THE UNIVERSITY
of EDINBURGH

Heuristic Methods for Solving Two Discrete Optimization Problems

José Xavier Cabezas García

Doctor of Philosophy
University of Edinburgh
2018

Declaration

I declare that this thesis was composed by myself and that the work contained therein is my own, except where explicitly stated otherwise in the text.

(*José Xavier Cabezas García*)

Acknowledgements

Firstly I thank God for all the good things that I have received during all the years of my life. I would like to say thanks to each of the people and institutions that have contributed to this achievement. I thank my supervisor Sergio García Quiles, who has helped me during all these years either in my research or in my daily concerns. He also raised my interest in location problems, which is currently one of the most important parts of my work. I wish to say thank you to my second supervisor Joerg Kalcsics with whom, alongside Sergio, we have shared several conferences and seminars.

My very special thanks to Julian Hall that without his help, support and encouragement, my arrival and adaptation to Scotland could have been much more difficult. The Hibs is my Scottish football team thanks to him. I thank Jacek Gondzio, head of our research group. I am particularly grateful for his support for the several academic activities that we have attempted to make, trying to contribute to the linear programming field. Thanks to Kerem Akartunali my external examiner for his valuable advices. I cannot neglect also to mention Gill Law for her invaluable help to me and other PhD students, she was always ready to help.

I am very thankful to the Government of Ecuador whom, through Senescyt (Secretaría Nacional de Ciencia y Tecnología) has supported my research with a generous scholarship. Furthermore, I wish to say thanks to ESPOL (Escuela Superior Politéctica del Litoral), my alma mater, that has also supported these years of work by a economic aid that permitted my family to join me in this journey.

I also want to thank all my friends that I have made during these years for the unforgettable moments that we spent together and that will be in my memory all my life: Wenyi, Minerva, Mook, Ivet, Basak, Marie, Marion, Jakub, Tom, Lukas, Rodrigo, Dominik, Nicolas, Robert and many others.

I have no words to say thanks to the people of this city. They are funny, kind and open minded. Edinburgh is such an organized city and culture is breathed everywhere. I am particularly thankful to my source of energy and well-being, Lian Pu, the best chinese restaurant in the city, highly recommended! Thanks Edinburgh, thanks Scotland.

Thanks to my family, here and in Guayaquil-Ecuador, for their continued unconditional support.

Thanks Daniela and Xavier Eduardo, my lovely kids, for behaving so well all these years. This has been a great experience for them. And thanks Lena for allowing me to follow my dreams with you. I love you.

Xavier Cabezas
Edinburgh, 2018

To my beloved family

Abstract

In this thesis we study two discrete optimization problems: Traffic Light Synchronization and Location with Customers Orderings. A widely used approach to solve the synchronization of traffic lights on transport networks is the maximization of the time during which cars start at one end of a street and can go to the other without stopping for a red light (bandwidth maximization). The mixed integer linear model found in the literature, named MAXBAND, can be solved by optimization solvers only for small instances. In this manuscript we review in detail all the constraints of the original linear model, including those that describe all the cyclic routes in the graph, and we generalize some bounds for integer variables which so far had been presented only for problems that do not consider cycles. Furthermore, we summarized the first systematic algorithm to solve a simpler version of the problem on a single street. We also propose a solution algorithm that uses Tabu Search and Variable Neighbourhood Search and we carry out a computational study. In addition we propose a linear formulation for the shortest path problem with traffic lights constraints (SPTL). On the other hand, the simple plant location problem with order (SPLPO) is a variant of the simple plant location problem (SPLP) where the customers have preferences on the facilities which will serve them. In particular, customers define their preferences by ranking each of the potential facilities. Even though the SPLP has been widely studied in the literature, the SPLPO has been studied much less and the size of the instances that can be solved is very limited. In this manuscript, we propose a heuristic that uses a Lagrangean relaxation output as a starting point of a semi-Lagrangean relaxation algorithm to find good feasible solutions (often the optimal solution). We also carry out a computational study to illustrate the good performance of our method. Last, we introduce the partial and stochastic versions of SPLPO and apply the Lagrangean algorithm proposed for the deterministic case to then show examples and results.

Lay Summary

This thesis focuses on two important discrete optimization problems: Traffic Lights Synchronization and The Simple Plant Location with Order. Traffic lights are primarily used in cities for the purpose of avoiding traffic accidents by controlling the flow of vehicles in motion. This allows pedestrians to cross the street while cars wait for the duration of a red light. The benefit of using traffic lights is clear, but its use also leads to problems, such as: Time delay when moving from one place to another, or increased pollution due to changes in the speeds of the vehicles. This problem is particularly difficult when studied on a network due to the inclusion of loops in the possible vehicles movements. The interval time where vehicles can pass through two corners of a single street without stopping for a traffic light (red time) is called bandwidth and it is preferable to be as large as possible. However it is not always possible to find a good bandwidth scheme on a whole network. The current more efficient methods prefer to use another measure to address the optimization, such as the total delay time of vehicles. In this work, we study the possibility to solve the bandwidth maximization on networks by a heuristic approach. We thoroughly study all the involved constraints and we give special attention to those involving loops.

Another very important issue for some firms in their business activities is to identify an optimal set of facilities that will be open and located in given possible sites in order to provide a service and cover the demands of a group of customers. It is possible that customers have preferences on what current open facilities will serve them. They can either rank the elements of the whole facilities set or only a subset of it. Furthermore, those preferences can be changed by the customers due to multiple circumstances, the order given can be considered a random variable that gives rise to a stochastic model. Clearly, the problem has two decisions to be made that can be seen as two decision stages: what facility will be open and the distribution of these to the customers. Studies about location without order have provided us a light about how to face the customer order case. These bases have allowed us to extend some classical results and adapt some solution schemes.

Contents

Chapter 1

Introduction

We start with a general introduction to this dissertation. We give our motivations and introduce the problems to be faced. The outline of the whole document is provided as well as the main contributions.

1.1 Two Important Discrete Problems

The users of a public or private service require a high level of quality and of course they expect that this has a price commensurate with its benefit. The main challenge for service suppliers is to create a product that considers all components that define the concept of quality for their customers, and at the same time be profitable for their interests. However, this is not an easy task as from the point of view of the consumer the value of a product depends on several variables; this includes service location (proximity), time of service, quality of the final product. Furthermore, service providers face problems arising from population growth, changing market developments or environmental demands, along with the fact that customers often expect service to improve over time. In this work, we would like to place ourselves within a context of win-win between suppliers and clients in problems with a strong impact on society and that until now continue to be challenging for researchers. We want to make this idea our main motivation.

This thesis focuses on two important discrete optimization problems: Traffic Lights Synchronization and The Simple Plant Location with Order. Traffic lights are primarily used in cities for the purpose of avoiding traffic accidents by controlling the flow of vehicles in motion. This allows pedestrians to cross the street while cars wait for the duration of a red light. The benefit of using traffic lights is clear, but its use also leads to problems, such as: Time delay when moving from one place to another, or increased pollution due to changes in the speeds of the vehicles. Our interest in this problem comes from the fact that it has been a subject of interest for several studies since the invention of traffic lights in the late 19th century, especially since urban traffic increased significantly after Henry Ford unveiled his Model T in 1908. This problem is particularly difficult when studied on a network due to the inclusion of loops in the possible vehicles movements. The interval time where vehicles can pass through two corners of a single street without stopping for a traffic light (red time) is called bandwidth and it is preferable to be as large as possible. However it is not always possible to find a good bandwidth scheme on a whole network, at least one that meets the logical requirements of all users of the transport network, since it is logical to assume that all users always want to go faster using the most comfortable route. The methodology used to solve this problem must consider this fact.

Another very important issue for some firms in their business activities is to identify an optimal set of facilities that will be open and located in given possible sites in order to provide

a service and cover the demands of a group of customers. It is possible that customers have preferences on what current open facilities will serve them. They can either rank the elements of the whole facilities set or only a subset of it. Furthermore, those preferences can be changed by the customers due to multiple circumstances, the order given can be considered a random variable that gives rise to a stochastic model. Clearly, the problem has two decisions to be made that can be seen as two decision stages: what facility will be open and the distribution of these to the customers. Studies about location without order have provided us a light about how to face the customer order case. These bases have allowed us to extend some classical results and adapt some solution schemes.

The following are general definitions of the mentioned problems:

Definition 1 (Traffic Light Synchronization Problem)**.** *The Traffic Light Synchronization Problem (TLSP) consists in the maximization on a network of the time during which cars start at one end of a street and can go to the other without stopping for a red light (bandwidth maximization).*

Definition 2 (Simple Plant Location Problem with Order)**.** *The Simple Plant Location Problem with order (SPLPO) is a variant of the simple plant location problem (SPLP) where the customers have preferences on the facilities which will serve them. In particular, customers define their preferences by ranking each of the potential facilities.*

Certainly these combinatorial problems are different in many aspects. Their modelling, for example, differ since TLSP is based on the intrinsic geometry of the transport networks, whilst SPLPO is concerned with this aspect only when it takes into account a cost measure of the distances between customers and suppliers. However, they both have things in common: Both are studied on a network and can be modelled as a linear integer program.

Our interest is in designing and implementing algorithms that solve both problems for instances that have not been solved so far in an approximate way. We base this on the fact that in both cases, large instances cannot be solved exactly in a reasonable time; indeed both problems are NP-hard problems.

The current most used and successful method for timing traffic lights is given by a commercial software which, rather than bandwidth, uses another measure to address the optimization, such as the total delay time of vehicles or the total fuel consumption. The software uses simulation to generate a possible traffic state and other components that allows it to emulate the behaviour of the network and evolutionary algorithms to improve an index of the performance measure used. Indeed, this method provides an approximate solution, i.e., this is a heuristic. Heuristic algorithms for bandwidth maximization have been less studied and, as far as we know, it is still a challenge for researchers. In this work, we study the possibility of solving the bandwidth maximization by a heuristic approach that uses algorithmic structures that have shown to be successful in other combinatorial problems. We also thoroughly study all the involved constraints and we give special attention to those involving loops.

On the other hand, even when large instances for the SPLP have been solved efficiently by exact methods, when customers' preferences are considered, small instances for this case become large instances for the SPLPO, as we noted in the literature. SPLP has been widely studied and a solution method usually applied involves the analysis of possible relaxations to the problem whose bounds are improved through iterative procedures. As it will be seen in later chapters, this approach can be found since the earliest papers related to this topic with good results. The contributions for the SPLPO are very few but of general interest for any future solution method. The most relevant ones try to strengthen the linear formulation of the problem by including new constraints that work as cuts of the feasible region and then use it in a branch and bound procedure. We seek to take advantage of the successful results obtained for the SPLP particularly with the use of Lagrangean relaxation and extend its use to the SPLPO.

We have been careful not to neglect the mathematics behind these two problems by proving theorems either related with different aspects of the modelling or related with the properties of the adapted procedures used in the design of the algorithms. Part of our contribution is indeed to prove that some results that have been used in more basic problems can be used in more general cases. In summary, the methods we propose are a mix of exact and heuristic approaches, all of them MILP (Mixed Integer Linear Programming) based algorithms. We tried to exploit this mathematical structure to apply approximation procedures which in turn rely on simple applications of either classical metaheuristic methods or simple variable fixing heuristics.

1.2 Research Questions

As mentioned above, the combinatorial nature of both problems suggests that heuristic methods are a valid and justified alternative solution. Looking at these methods, we find in the literature few research related to TLSP (bandwidth maximization) and SPLPO. However, some studies using MILP-based algorithms for the TLSP and for the version without customer preferences of the SPLPO are known. Under this scenario, the following research questions emerge: How can the results previously obtained in either similar problems or less restricted problems be extended to those studied in this work? Are MILP-based heuristic algorithms to these two problems comparable to the established commercial software MILP algorithms? The contribution of this work will depend on whether these questions are answered successfully.

1.3 Contributions

Some results for TLSP and SPLP have been extended to more general cases. In TLSP we give bounds over integer variables used in its linear formulation. They are based in others given when the problem is solved over a single avenue. Likewise, we were able to prove that results obtained for SPLP in the area of Lagrangean and semi-Lagrangean relaxation can be extended to SPLPO. Making use of this we developed algorithms for both problems addressed. They are within a heuristic class of procedures that use either linear programming, fixing variables or both. Numerical experiments have been carried out to verify their performance in larger instances than those found in the literature.

Additionally, we introduce a novel formulation for the Shortest Path Problem with Traffic Light Constraints (SPPTL). This problem has been solved before by using a variation of a known labelled-based algorithm but, as far as we know, no linear models have been defined. Our model uses the flow-based formulation for the Shortest Path Problem (SPP) as a template and the traffic light behaviour is modelled by periodic time windows. Also, since customers in SPLPO can have preferences over a subset of facilities instead all of them we introduced a linear formulation for the Simple Plant Location Problem with Partial Order (SPLPPO). Furthermore a stochastic version of SPLPO was studied by considering the order given by the customers as an uncertain event that can be modelled by scenarios. A Two-Stage stochastic model for this case (S-SPLPPO) is given.

1.4 Outline of the Thesis

Since we deal with two problems that differ in most of their characteristics, this thesis can be read in two parts independently.

Part I: Traffic Light Synchronization

Chapter 2: The Traffic Light Synchronization Problem

First we provide the main concepts of graph theory that will be used in this chapter as well as an algebraic treatment of cycle basis that are important in the modelling of TLSP. In Section 2.3 we review MAXBAND, a bandwidth maximization model for TLSP. We give a detailed explanation of all its elements and introduce the notation used. Particularly, in Section 2.3.4 we show how to model the loop constraints, presented in the MAXBAND linear model, with a small number of binary variables. In Section 2.4 we generalize the bounds for integer variables that can be found on the literature. The first known systematic algorithm for the TLSP without loops is reviewed in Section 2.5. In Section 2.6 we propose a new MILP-based heuristic algorithm using tabu search and variable neighbourhood search. We carry out a computational study to show that our method performs very well for large instances. Some conclusions and ideas for future research are discussed in Section 2.6.2. Finally, in Section 3 we propose a linear model for the Shortest Path Problem with Traffic Light Constraints (SPPTL).

Chapter 3: An MILP model for a Related Problem to TLSP

There are other related problems where traffic lights have an important role, one of them is studied in Section 3.1. We defined the notation for the SPPTL in Section 3.1.1 to then introduce a MILP model in Section 3.2. We tested the proposed model in Section 3.3 by running an example that helps to understand the notation better. In Section 3.4 we give final conclusions and remarks.

Part II: Simple Plant Location Problem with Order

Chapter 4: A Lagrangean Relaxation Algorithm for the SPLPO

This chapter starts with Section 4.2.1 where we review Lagrangean and semi-Lagrangean relaxation. The SPLPO version of those models are shown in Section 4.3, 4.3.1, 4.4 and 4.4.1, along with the methods that will be used later to solve their respective dual problems. The complete procedure that we propose to solve the SPLPO is presented in Section 4.5, where all the algorithms given in previous sections will be gathered to build a heuristic procedure that uses few parameters to be set up. In Section 4.6 the whole method is tested over a group of large instances. Some conclusions are given in Section 4.7.

Chapter 5: The Stochastic Simple Plant Location Problem with Partial Order

In Section 5.3.1 we defined the Simple Plant Location Problem with Partial Order (SPLPPO)and discussed its linear formulation. Then, we present a two-stage stochastic version of this problem (S-SPLPPO) in Section 5.4 and finally some experiments are carried out in Section 5.5.

Part I

Traffic Light Synchronization

Chapter 2

The Traffic Light Synchronization Problem

The main goal of this chapter is threefold. First we establish the theoretical framework that we will use throughout this part of the document. We give an brief literature review of traffic light synchronization problem, introduce the concept of bandwidth maximization and define a mixed integer linear formulation that models the problem on a transport network. Second we study a classical algorithm to solve the problem along a single artery that does not consider any turns to any other street. Finally we address the more general case and propose an algorithm to solve it. With this algorithm we try to take advantage of the linear formulation by taking into account bounds on the integer variables in a fixing variable iterative procedure with memory structure.

2.1 Introduction to Traffic Lights Synchronization

Traffic lights have been with us for a long time (since late in the 19th century) and they are used in cities to control the flow of vehicles. But its use also leads to some problems such as time delays when moving from one place to another and increased pollution due to changes in the speeds of the vehicles. Because of the increase in urban traffic year by year, its timing has been a topic of interest for many researchers. Most of the papers focus on two aspects: minimizing some measure to assess the performance of the traffic (e.g, delays) and maximizing the time that vehicles can drive without stopping for red lights (bandwidth maximization).

With respect to traffic flow measures, it has been usual to minimize either the overall delay or the number of stops of the vehicles. One of the first models was developed by Gartner et al. (1975). This study demonstrated the feasibility of using mixed integer linear programming to optimize traffic signal settings for practical but small size road networks. In that work a convex nonlinear objective function and linear constraints were considered, but a piecewise linearisation in the objective can be done by increasing the number of constraints in the formulation. The objective function measures the overflow queue wich represents the number of vehicles that are not able to clear an intersection during the preceding green time and the constraints consider the possible loops in which a platoon of vehicles could incur when navigating in a network. However, some realistic constraints were not considered in this case. As an example, different patterns of light changes at street junctions were not taken into account; this would allow to consider either crosswalk times or the synchronization of traffic lights towards cross streets. This method was also used by Wünsch (2008) to show a similar linear model with some differences. One is that the model is able to decide among different predetermined signal timing plans. Another is that the assumption of a common red-green period width at the signals is relaxed. However, it is not

a hard constraint because in that case the least common multiple of all red-green periods could be used as a uniform period, as mentioned in Köhler and Strehler (2015). The linear model was tested on a family of real-world transport networks of up to 146 nodes and 399 arcs. Also Wünsch (2008) showed a formal proof of the NP-completeness of the signals timing problem. Another reference can be found in Improta and Sforza (1982), the authors defined a linear model similar to the one proposed by Gartner et al. (1975) and additionally developed a branch and bound procedure with backtracking to solve it which relaxes some assumptions imposed in the original model. Unfortunately, only small examples are reported (up to 9 nodes and 16 arcs).

With regard to bandwidth maximization, one of the first papers was Morgan and Little (1964). In that paper, the authors presented a geometric and intuitive method for bandwidth maximization on a two-way street with a given fixed and common red-green time period on each signal and preassigned vehicle velocities. Even though there had been some geometric methods developed earlier, this paper presented a systematic algorithm easy to implement. A couple of years later, Little (1966) proposed for the first time a mixed integer linear program (MILP) to solve a new version of the problem (more complete) which does not assume a fixed red-green time period, but this is chosen by the model between some given bounds. Upper and lower limits on velocity between adjacent signals and changes in speed are also considered. Solving the model yields a common signal red-green period, velocities between signals and maximal bandwidths on the streets. An extension is provided to solve the same problem on general networks, but only very small instances could be solved. On networks the problem is more difficult because it is necessary to introduce the so-called loop constraints, which permit to model the circular movements that vehicles can do. Some years later, Little et al. (1981) introduced a generalization that includes left turns at junctions. This linear model is traditionally called MAXBAND even though this is the name of the code developed to solve it. The first version of MAXBAND could handle problems on networks with only 3 arteries and up to 17 traffic signals. Later Gartner et al. (1991) extended the MAXBAND model by working with variable bandwidth for each street segment (MULTIBAND). This change allowed to incorporate a traffic factor on the objective function. More recently Zhang et al. (2015) proposed a new version of MULTIBAND called AM-BAND. This model tries to use better the available green times on both road directions by relaxing a symmetric assumption with respect to the progression line given in the MULTIBAND linear model. In Xianyu et al. (2012) and Xianyu et al. (2013) a variable bandwidths is also considered, their approach uses the criteria of partitioning a large system into smaller subsystems and takes into account the impact of speed variation. However, this method is not based on integer programming.

The use of heuristics to solve traffic light synchronization problems is common, specially due to the influence of the very successful commercial software for synchronization of traffic signals and traffic management named TRANSYT (Cohen, 1983), which is an implementation of a method introduced by Robertson (1969). TRANSYT provides a heuristic solution for a very complete and robust objective function by using microscopic simulation of traffic behaviour and genetic algorithms. Other references of using evolutionary methods can be seen in Braun and Weichenmeier (2005) and Singh et al. (2009). Gartner and Stamatiadis (2002) proposed a heuristic method for the MAXBAND network problem that in a first stage solves a tree subproblem which considers a measure of interest. Then, the integer variables are fixed to the values obtained and used in a second stage to solve the whole problem. As far as we know there is no other reference about solving the MAXBAND on a complete network.

Since MAXBAND can only be solved by optimization solvers for very small instances, the main aim of this work is to develop a heuristic algorithm for bandwidth maximization. As the algorithm is based on the MAXBAND formulation, first we review the constraints in detail. Later, we generalize some existing bounds based on the ideas outlined in Little (1966). Then,

we propose a metaheuristic algorithm to solve the problem on complete networks. We carry out some computational experiments to verify how efficient the method is. The method we propose starts with a feasible solution of the problem and uses basic Tabu Search (Glover, 1986) ideas such as the use of a memory structure within an iterative local search process that allows that solutions found during execution to be more diverse. In addition, the search is intensified by a sequence of neighbour solutions search processes, just as in Variable Neighbourhood Search (VNS), see Mladenović and Hansen (1997).

One of the key factors that influence the performance of any of the methods mentioned above is the large amount of information they require from the network. Transport networks in the real world are, in some cases, similar to a grid of streets that intersect with each other (grid graph). Therefore, each intersection may have traffic lights for possible vehicle entrances from four different directions. Information of the red and green light times on each of these can be required, as is the case of MAXBAND. Additional data such as the length of the queue of vehicles that wait to cross or turn to another street are also necessary. If real data is not available, a simulation of the network information must be carried out. The parameters to be generated must be consistent to allow feasible solutions; for example, green lights at the same time can not be allowed for all signals that are in the same junction of streets. Unfortunately, we have not been able to access real data and therefore we have opted to simulate them.

The rest of the chapter is structured as follows. First we provide the main concepts of graph theory that will be used in this chapter as well as an algebraic treatment of cycle basis which will be used in the modelling of TLSP. Then, we review the MAXBAND model in Section 2.3, where we give a detailed explanation of all its elements and introduce the notation that will be used. Particularly, in Section 2.3.4 we show how to model the loop constraints with a small number of binary variables. In Section 2.4 we generalize the bounds for arterial integer variables that were originally presented by Little (1966) for graphs that do not consider loops and extend them for general networks. The first known systematic algorithm for TLSP without loops is reviewed in Section 2.5, since this method gives us a deep understanding of the geometry related to the formulation of MAXBAND for the case without loops. In Section 2.6 we propose a new MILP-based heuristic algorithm that uses Tabu Search and Variable Neighbourhood search. We carry out a computational study to show that our method performs very well for large instances when an initial feasible solution is provided. Some conclusions and ideas for future research are discussed in Section 2.6.2.

2.2 Preliminaries

In this section we will review fundamental concepts of cycle bases which are important to the TLSP modelling, as can be seen later. The notation and theoretical basis used are based on Diestel (2000) and Kavitha et al. (2009).

Undirected Graph

A *Finite Undirected Graph* is a pair $G = (V, E)$ where V is a finite set and E is a finite family of pairs of elements of V. The elements of V are called *vertices* or *nodes* (more commonly *vertices*) and the elements of E are called *edges*. $V(G)$ and $E(G)$ will be the notations to specify that V and E correspond to a specified graph G. A pair $e = \{u, v\} \in E(G)$ can appears more than one time in $E(G)$ (this justifies use the term *family* to refer to set of edges as is pointed in Schrijver (1986)), in this case e is called a *multiple edge*. A *loop* is an edge $\{e, e\}$. If a graph doesn't have multiple edges or loops it is called a *simple graph*. u and v in $e = \{u, v\}$ are called *ends* and it is said e is *incident* to u and v. Frequently we use uv to refer to edge $\{u, v\}$. The *degree* of a

vertex v is noted by $\delta(v)$ and represents the number of the edges incidents to v if there is not a loop vv, otherwise the loop counts twice.

The union and intersection between two graphs $G = (V, E)$ and $G' = (V', E')$ are set as $G \cup G' = (V \cup V', E \cup E')$ and $G \cap G' = (V \cap V', E \cap E')$. $G \setminus G'$ is the *difference* between G and G' and results form removing the vertices $V(G) \cap V(G')$ and their incidents edges from G. If $V \subset V'$ and $E \subset E'$ we say $G \subset G'$ also G is a *subgraph* of G'. An *induced graph* de G is an subgraph that is the result of removing a subset of vertices of $V(G)$ and the edges incidents to them.

A *path* in a graph $G = (V, E)$ is a subgraph P of G where $V(P) = \{v_0, v_1, \ldots, v_k\}$ and $E(P) = \{v_o v_1, v_1 v_2, \ldots, v_{k-1} v_k\}$, the notation $P(v_0, v_k)$ is used to call the path from v_0 to v_k. The *length* of a path is the number of edges in the path. A path $P(v_0, v_k)$ is *closed* if $v_k = v_0$. The graph $G = (V, E)$ is *connected* if there is at least a path between any pair of vertices, otherwise is *disconnected*. If a graph is disconnected then it is formed by connected subgraphs, each one of them is called *connected component* of G. If a graph T is connected and has no cycles it is called a *tree*. A *forest* is an undirected graph in which all of its connected components are trees. An spanning tree T of an undirected graph G is a subgraph of G that is a tree which includes all of the vertices of G.

A *cycle* ζ in G is a subgraph of a graph G with even $\delta(v)$ for all $v \in \zeta$. Just as in Kavitha et al. (2009), we will call a *circuit* if it is a connected cycle and each one of its vertices has degree two, i.e if it is a simple connected cycle.

Directed Graph

A *Finite Directed Graph* (*finite digraph*) is a pair $D = (V, A)$ where V is a finite set and A is a finite family of *ordered* pairs of elements of V. Again the elements of V are called *vertices* or *nodes* (more commonly *nodes*) and the elements of A are called *edges* or *arcs* (more commonly *arcs*). In the pair $e = (u, v) \in A(D)$, u and v are refereed to as *tail* and *head* respectively, it makes sense to represent sometimes (u, v) as $u \to v$. The number of arcs of the form (u, v) is called *indegree* $\delta^-(v)$ of the vertex v, and the number of arcs of the form (v, t) *outdegree* $\delta^+(v)$. A *directed path* $Pd(v_0, v_k)$ of a directed graph $D = (V, A)$ is a subgraph Pd of D where $V(Pd) = \{v_0, v_1, \ldots, v_k\}$ and $A(Pd) = \{(v_o, v_1), (v_1, v_2), \ldots, (v_{k-1}, v_k)\}$. Analogously to the case undirected the *length* of a directed path is the number of the arcs that forms the path.

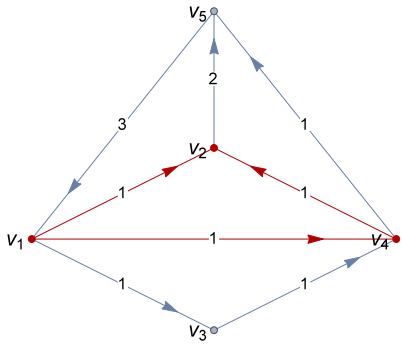
Every directed graph has an *underline* graph which is undirected, taking the directions on the edges out. Analogously all undirected graph becomes a directed one if it is added up an arbitrary *orientation* on every edge in the graph. There are undirected paths $P(v_0, v_k)$ on a directed graph $D = (V, E)$ with edges (arcs) with two possible orientations *forward* or *backward*, in the same way D may have also undirected cycles. A *directed cycle* is then a cycle where every edge can be either forward or backward. In this context we can also define *directed spanning tree*.

κ-cycle

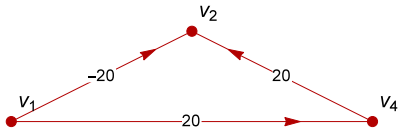
In order to follow to Liebchen and Rizzi (2007) and Kavitha et al. (2009) and to give a more algebraic treatment to directed cycles we provide the next definition. A *κ-cycle* ζ over a field κ is a subgraph of a directed graph $D = (V, A)$ that can be represented by a vector $\zeta = [\zeta(1), \zeta(2), \ldots, \zeta(|A(\zeta)|)]^T \in \kappa^{|A(\zeta)|}$ such that $\sum_{e \in \delta^+(v)} \zeta(e) = \sum_{e \in \delta^-(v)} \zeta(e)$ for all $v \in V(D)$ (*flow conservation*), κ-cycles sometimes are called *circulations*.

It is commonly assumed that $\kappa = \mathbb{Q}$, then we can assign a positive value to $\zeta(e)$ if e is forward and a negative value if e is backward in a subgraph of D. If this assignment is possible such that the subgraph becomes a circulation, then we have a $\mathbb{Q}$-cycle. A *simple κ-cycle* is one

represented by a vector with every component in the set $\{-1, 0, 1\}$ (if the field allows it). It makes sense if it is decided that the values of the field κ represent the times that an arc can be repeated if we "circulate" from a node to another (this means all flows have integer values). If a simple κ-cycle is connected and every vertex has just two incident arcs it is called *circuit*, which "geometrically" looks like a connected directed cycle, see Figure 2.1b.



(a) A circulation with 5 nodes.



(b) Circulation (no simple) with all nodes visited 20 times.

Figure 2.1: Circulations of 5 nodes over $\kappa = \mathbb{Q}$ and a no simple cycle subgraph.

If in Figure 2.1b we change the flow over the edges $(v_1, v_2), (v_4, v_2)$, and (v_1, v_4) by -1,1 and 1 respectively, it becomes a circuit (a directed cycle).

Let us consider a $\mathbb{Q}$-cycle ζ with integral components. Then $\pi(\zeta) = \zeta(e)\ mod\ 2$ will be the *projection* of ζ with entries in the field $\kappa = \mathbb{F}_2 = \{0, 1\}$, this means it becomes an $\mathbb{F}_2$-cycle. An $\mathbb{F}_2$-cycle is a good representation for an undirected cycle of the underline undirected graph D' of the directed graph $G = (V, A)$.

In the next section we show in more detail the algebra of $\mathbb{F}_2$-cycles.

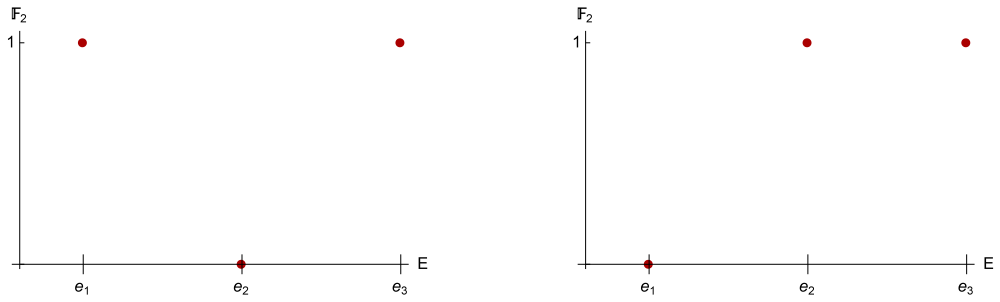
Cycle basis on undirected graphs

As is mentioned in Kavitha et al. (2009) the study of cycle basis dates back to the early days of graph theory, as an example MacLane (1937) gave a characterization of planar graphs in terms of Cycle basis. However, we will show results given by Kavitha et al. (2009) and Schrijver (1986) as these are simpler and understandable.

Let $G = (V, E)$ be a graph where V represents a set of vertices $\{v_1, v_2, \ldots, v_n\}$ with $|V| = n$ and E represents a set of edges $\{e_1, e_2, \ldots, e_m\}$ with $|E| = m$. It is possible to define a vector space (vertex space) $\mathcal{V}(G)$ on the vertices of G and the field $\mathbb{F}_2$. Let $\mathcal{V}(G)$ be the set of all functions of the form $\mathcal{V} : V \to \mathbb{F}_2$, this means that $\mathcal{V}(G)$ is the power set of V, because any of these functions maps vertices to numbers 0 or 1 and thinking the elements of $\mathbb{F}_2$ like values of an indicator variable. The operations $+ : \mathcal{V}(G) \times \mathcal{V}(G) \to \mathcal{V}(G)$ and $\cdot : \mathbb{F}_2 \times \mathcal{V}(G) \to \mathcal{V}(G)$ are defined too, such that $\nu_1 + \nu_2 = \nu_1 \triangle \nu_2$ where $\triangle$ represents the symmetric difference $(\nu_1 \backslash \nu_2) \cup (\nu_2 \backslash \nu_1)$. Also, $k \cdot \nu = \nu$ if $k = 1$ and $k \cdot \nu = \varnothing$ if $k = 0$. It is clear that the neutral element is $\varnothing$ and that for each $\nu \in \mathcal{V}(G)$ its inverse element is the same ν, therefore $\nu = -\nu$. A basis of $\mathcal{V}(G)$, actually a canonical basis, will then be $\{\{v_1\}, \{v_2\}, \ldots, \{v_n\}\}$ and therefore $dim\mathcal{V}(G) = |V| = n$.

In the same way an edge space $\mathcal{E}(G)$ of all functions of the form $\mathcal{E} : E \to \mathbb{F}_2$ can be defined, under the same field $\mathbb{F}_2$ and under the same closed operations $+ : \mathcal{E}(G) \times \mathcal{E}(G) \to \mathcal{E}(G)$, $\varepsilon_1 + \varepsilon_2 = \varepsilon_1 \triangle \varepsilon_2$ and $\cdot : \mathbb{F}_2 \times \mathcal{E}(G) \to \mathcal{E}(G)$, $k \cdot \varepsilon \in \{\varepsilon, \varnothing\}$ as above. $\mathcal{E}(G)$ is then the

power set of E. Furthermore a canonical basis of this space is $\{\{e_1\},\{e_2\},\ldots,\{e_m\}\}$ and $dim\mathcal{E}(G) = |E| = m$.



(a) $\varepsilon_1 = \{e_1, e_3\}$ (b) $\varepsilon_2 = \{e_2, e_3\}$

Figure 2.2: Example of two functions in $\mathcal{E}(G)$ with $\varepsilon_1 + \varepsilon_2 = \varepsilon_1 \triangle \varepsilon_2 = \{e_1, e_2\}$.

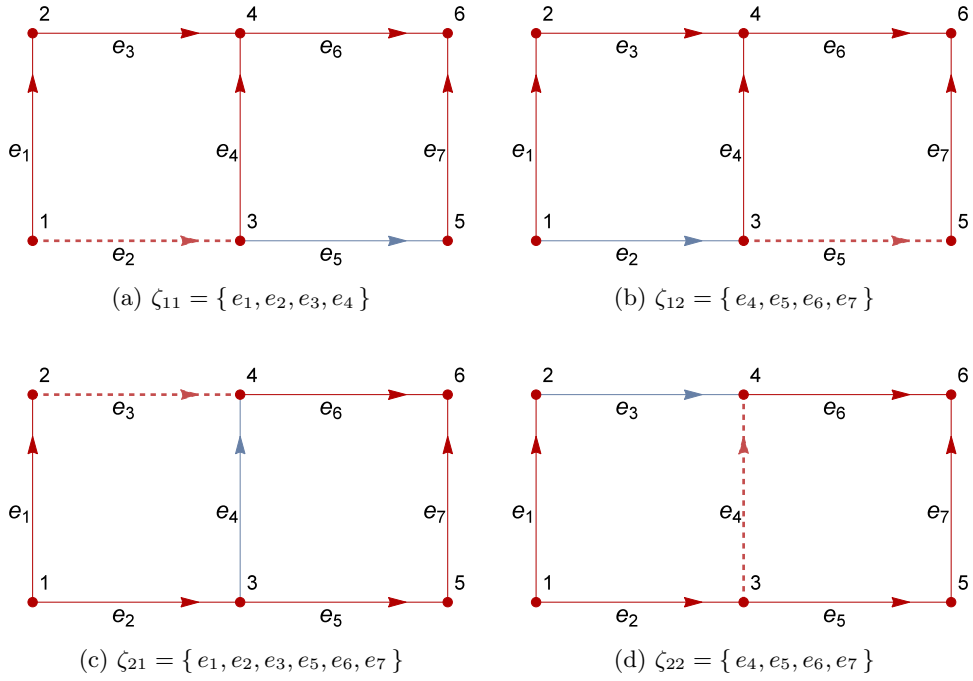
Let the column array $[k_1, k_2, \ldots, k_m]^T$ be the coordinate of the vector $\varepsilon \in \mathcal{E}(G)$ on the canonical basis, i.e., $[\varepsilon]_{B(\mathcal{E}(G))} = [k_1, k_2, \ldots, k_m]^T$ whenever $\varepsilon = k_1\{e_1\} + k_2\{e_2\} + \ldots + k_m\{e_m\}$. Now it is possible to define an inner product between two vectors in $\mathcal{E}(G)$ as $\langle \varepsilon_1, \varepsilon_2 \rangle = [\varepsilon_1]^T_{B(\mathcal{E}(G))}[\varepsilon_2]_{B(\mathcal{E}(G))} = k_1^1 k_1^2 + k_2^1 k_2^2 + \ldots + k_m^1 k_m^2$. It is not a proper inner product because $\langle \varepsilon, \varepsilon \rangle$ can be zero even though $\varepsilon \neq \varnothing$, which is why it is called indefinite inner product, see Hotovy et al. (2015). According to the example of Figure 2.2, the inner product between $\langle \varepsilon_1, \varepsilon_2 \rangle = [1,0,1][0,1,1]^T = 0+0+1 = 1$, but $\langle \varepsilon_2, \varepsilon_2 \rangle = [0,1,1][0,1,1]^T = 0+1+1 = 0$ even when ε_2 is different from the zero vector $\varnothing$. It can be seen that $\langle \varepsilon_1, \varepsilon_2 \rangle = 0 \in \mathbb{F}_2$ if and only if ε_1 and ε_2 have an even number of edges in common.

There is one important subspace of $\mathcal{E}(G)$ to be considered, the *cycle space* $\mathcal{C}(G)$ which contains all the cycles in G. On a connected graph $G = (V, E)$ with a spanning tree T we can see an important property. It is true if an arc that is not in the tree is added, this yields one and only one cycle, see Figure 2.3. That cycle is called *Fundamental Cycle* with respect to T and by adding one by one the arcs in $G \backslash T$ it results in a set of fundamental cycles that spans the Cycle Space $\mathcal{C}(G)$ that is linearly independent.

Theorem 1 (Kavitha et al. (2009))**.** *The set of fundamental cycles of a connected graph $G = (V, E)$ with a spanning tree T, $|V| = n$ and $|E| = m$ is a basis for the cycle space $\mathcal{C}(G)$ of length $m - n + 1$.*

Proof. By definition and construction of fundamental cycles with respect to a fix spanning tree T none of them can be obtained as linear combinations of the others, it is because each fundamental cycle ζ_i of G will have one edge e that is not in any other fundamental cycle ζ_j. That edge e is of course one which forms the cycle and belongs to $G \backslash T$. This proves the independence. Furthermore, since any spanning tree T in G has $n-1$ edges there are $m-(n-1)$ e's to form fundamental cycles.

Let ζ_i be a fundamental cycle in G with respect to T, e_i the edge in $G \backslash T$ that forms ζ_i and ζ an arbitrary cycle in $\mathcal{C}(G)$. Then $k_1\zeta_1 + k_2\zeta_2 + \ldots + k_{m-n+1}\zeta_{m-n+1} = \zeta$. Indeed it happens because each ζ contains one edge that is not in T and that does not belong to any other cycle, then the sum $k_1\zeta_1 + k_2\zeta_2 + \ldots + k_{m-n+1}\zeta_{m-n+1} + \zeta = \varnothing$, which is also a cycle, contains only arcs in T, and it means that the right hand of the equation is an acyclic graph and the only acyclic cycle in $\mathcal{C}(G)$ is $\varnothing$. ∎



(a) $\zeta_{11} = \{ e_1, e_2, e_3, e_4 \}$ (b) $\zeta_{12} = \{ e_4, e_5, e_6, e_7 \}$

(c) $\zeta_{21} = \{ e_1, e_2, e_3, e_5, e_6, e_7 \}$ (d) $\zeta_{22} = \{ e_4, e_5, e_6, e_7 \}$

Figure 2.3: Two Spanning Tree and its Fundamental Cycles.

Figure 2.3 shows an example with two different spanning trees, T_1 in Figure 2.3a and 2.3b, and T_2 in Figure 2.3c and 2.3d. The basis of the cycle space $\mathcal{C}(G)$ with respect to T_1 is $\{ \zeta_{11}, \zeta_{12} \}$ and with respect to T_2 is $\{ \zeta_{21}, \zeta_{22} \}$.

2.3 The MAXBAND Model for a Network

Now we define the MAXBAND model notation and give a complete explanation of all variables and constraints involved. Particularly, we discuss in detail how loops are modelled.

Let us consider a group of two-way *arteries* (streets) that meet each other in *junctions* forming a transport *network*. This network has some traffic signals in order to regulate traffic. They work with a common *period* that is split in red and green times, this will be used as unit of measurement of time. It is sometimes called *cycle length*, although we will try to avoid the use of this term because it can be confused with the *loops* (also known as cycles) that are present in the network and which play a very important role in the formulation of the problem, as we will see later. Instead, we will use the name *period length*. The distances (time units) that allow to measure the relative location between two signals on the same artery and on different arteries are called *internode offset* and *intranode offset* respectively. A list of offsets for the signals is said to be a *synchronization*.

The MAXBAND model is based on the geometry that we can see in Figure 2.4. Information is provided for two signals S_{ai} and S_{aj} on an artery a. The notation used is essentially the same than in Little et al. (1981). The different variables and parameters are defined as follows:

Parameters in Figure 2.4:

- T: Period length, in seconds.
- n_a: Number of traffic lights (signals) on artery a.
- r_{ai} ($\overline{r}_{ai}$): Outbound (inbound) red time of signal i on artery a, in a fraction of the period.
- τ_{ai} ($\overline{\tau}_{ai}$): An advancement of the outbound (inbound) bandwidth upon leaving S_i, in a fraction of the period.

Variables in Figure 2.4:

- z: Signal frequency, in periods per second.
- b_a ($\overline{b}_a$): Outbound (inbound) bandwidth on artery a, in a fraction of the period.
- t^a_{ij} ($\overline{t}^a_{ij}$): Travel time from S_{ai} to S_{aj} in outbound (from S_{aj} to S_{ai} in inbound) direction, in periods.
- ϕ^a_{ij} ($\overline{\phi}^a_{ij}$): Time from the center of red at S_{ai} to the center of red at S_{aj}, in periods. The two reds are chosen so that each is immediately to the left (right) of the same outbound (inbound) green band. ϕ^a_{ij} ($\overline{\phi}^a_{ij}$) is positive if S_{aj}'s center of red lies to the right (left) of S_{ai}'s.
- w_{ai} ($\overline{w}_{ai}$): Time from the right (left) side of S_{ai}'s red to the left (right) side of green band in outbound (inbound) direction, in a fraction of the period.
- Δ_{ai} : Time from center of $\overline{r}_{ai}$ to the nearest center of r_{ai}, in periods. It is positive from left to right.

In Figure 2.4 a two direction (outbound and inbound) street (artery) is depicted in a space-time plane. The continuous lines formed by the union of red and green segments (a period), shown above the signals S_{ai} and S_{aj}, represent the periodic behaviour of the traffic lights over time in the outbound direction, while the dash lines correspond to inbound. On each signal both lines do not necessarily overlap. This makes sense since it is possible to consider different movement patterns of vehicles to cross streets from the considered artery and vice versa, as we will see in Section 2.3.2. If these schemes are not taken into account both lines will coincide, as is the case of long roads where the red lights must be present only to allow the crossing of pedestrians in certain points of the road. Points A, B, C and D will be useful to define the different relationships between the variables and parameters necessary for the formulation.

The complete formulation for MAXBAND on a network can be seen in LM 2.3.1. It has additional parameters and variables (not showed in Figure 2.4), jointly with the constraints, these are explained next. Similar to Little et al. (1981), in this thesis we do not write $\alpha^a_{i,i+1}$ but α^a_i for any parameter or variable α and their corresponding names with bars for the opposite direction on an artery.

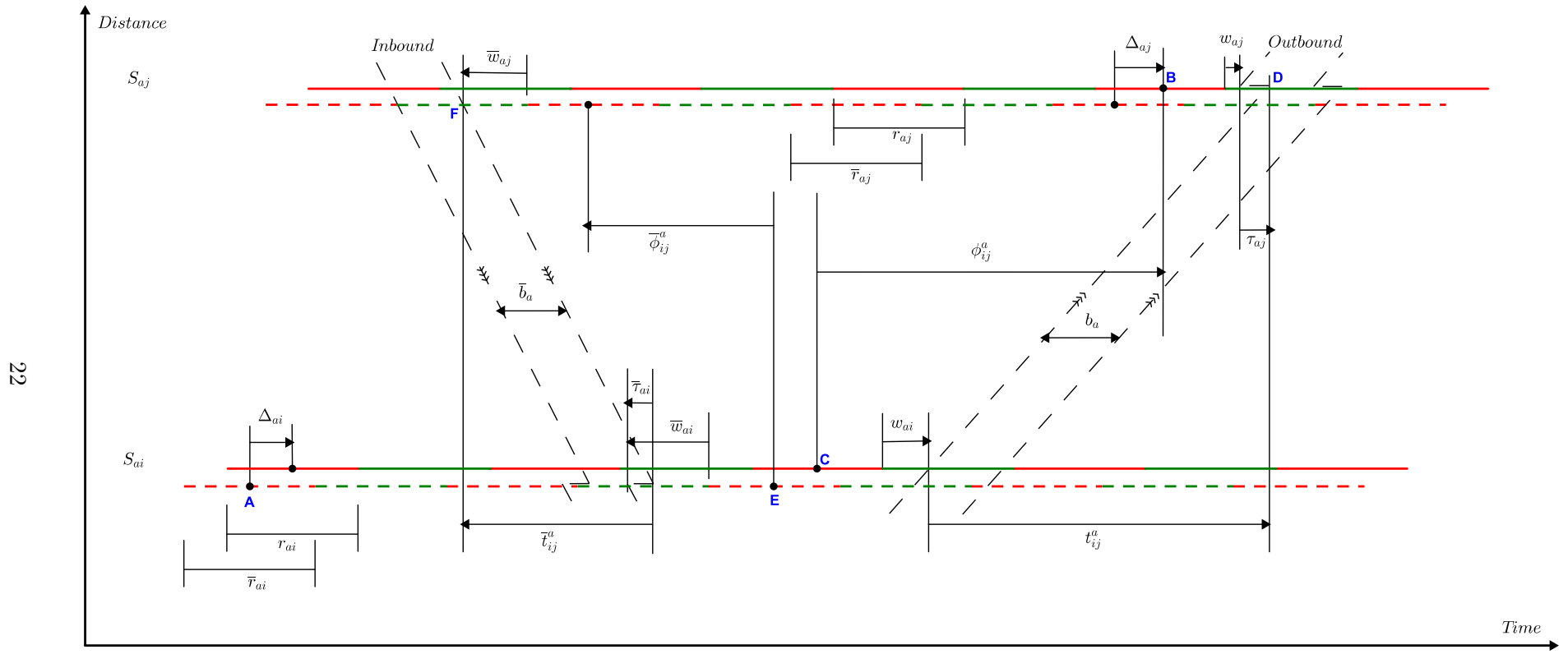


Figure 2.4: Geometry for MAXBAND model on artery a.

LM 2.3.1 MAXBAND, Maximal Bandwidth Formulation

Maximize
$$\sum_{a\in A}\left(k_a b_a + \overline{k}_a \overline{b}_a\right) \tag{2.1}$$

subject to :

$$\frac{1}{T_2} \le z \le \frac{1}{T_1}, \tag{2.2}$$
$$w_{ai} + b_a \le 1 - r_{ai}, \qquad \forall a \in A,\ \forall i = 1,\dots,n_a, \tag{2.3}$$
$$\overline{w}_{ai} + \overline{b}_a \le 1 - \overline{r}_{ai}, \qquad \forall a \in A,\ \forall i = 1,\dots,n_a, \tag{2.4}$$
$$\begin{aligned}&(w_{ai} + \overline{w}_{ai}) - (w_{a,i+1} + \overline{w}_{a,i+1}) + (t_i^a + \overline{t}_i^a)\\ &+(\delta_{ai}\ell_{ai} - \overline{\delta}_{ai}\overline{\ell}_{ai}) - (\delta_{a,i+1}\ell_{a,i+1} - \overline{\delta}_{a,i+1}\overline{\ell}_{a,i+1}) + (r_{ai} - r_{a,i+1})\\ &-(\tau_{a,i+1} + \overline{\tau}_{ai}) = m_i^a, \qquad \forall a \in A,\ \forall i = 1,\dots,n_a - 1,\end{aligned} \tag{2.5}$$
$$\left(\frac{d_i^a}{f_i^a}\right) z \le t_i^a \le \left(\frac{d_i^a}{e_i^a}\right) z, \qquad \forall a \in A,\ \forall i = 1,\dots,n_a - 1, \tag{2.6}$$
$$\left(\frac{\overline{d}_i^a}{\overline{f}_i^a}\right) z \le \overline{t}_i^a \le \left(\frac{\overline{d}_i^a}{\overline{e}_i^a}\right) z, \qquad \forall a \in A,\ \forall i = 1,\dots,n_a - 1, \tag{2.7}$$
$$\left(\frac{d_i^a}{h_i^a}\right) z \le \left(\frac{d_i^a}{d_{i+1}^a}\right) t_{i+1}^a - t_i^a \le \left(\frac{d_i^a}{g_i^a}\right) z, \qquad \forall a \in A,\ \forall i = 1,\dots,n_a - 2, \tag{2.8}$$
$$\left(\frac{\overline{d}_i^a}{\overline{h}_i^a}\right) z \le \left(\frac{\overline{d}_i^a}{\overline{d}_{i+1}^a}\right) \overline{t}_{i+1}^a - \overline{t}_i^a \le \left(\frac{\overline{d}_i^a}{\overline{g}_i^a}\right) z, \qquad \forall a \in A,\ \forall i = 1,\dots,n_a - 2, \tag{2.9}$$
$$\sum_{(i,j):a\in A_\zeta^F} \phi_{ij}^a \;-\; \sum_{(i,j):a\in A_\zeta^B} \phi_{ij}^a \;+\; \sum_{(b,j,i,c,k)\in J_\zeta} \Psi_{S_{bj},S_{ck}}^i = C_\zeta, \qquad \forall \zeta \in \mathcal{B}_\zeta, \tag{2.10}$$
$$C_\zeta \in \mathbb{Z}, \qquad \forall \zeta \in \mathcal{B}_\zeta, \tag{2.11}$$
$$m_i^a \in \mathbb{Z}, \qquad \forall a \in A,\ \forall i = 1,\dots,n_a - 1, \tag{2.12}$$
$$\delta_{ai}, \overline{\delta}_{ai} \in \{0,1\}, \qquad \forall a \in A,\ \forall i = 1,\dots,n_a - 1, \tag{2.13}$$
$$b_a, \overline{b}_a, t_i^a, \overline{t}_i^a, w_{ai}, \overline{w}_{ai}, z \ge 0, \qquad \forall a \in A,\ \forall i = 1,\dots,n_a - 1. \tag{2.14}$$

Furthermore, we consider only networks that can be represented by two dimensional grid graphs $G_{r\times c}(V,E)$, where r and c are the number of rows and columns, respectively. V is the set of nodes, E is the set of edges, $|V| = n = rc$ and $|E| = m = 2rc - r - c$. Grid graphs are a good representation of many real-world networks. An example is shown in Figure 2.5 with $n = 12$ and $m = 17$.

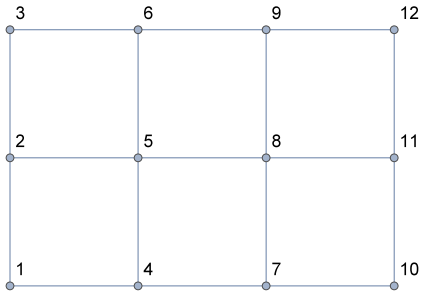


Figure 2.5: A grid graph $G_{3\times 4}(V,E)$.

2.3.1 Objective Function

In terms of level of vehicular congestion, in real urban networks some streets or avenues can be more important than others. If this is the case, these important arteries must be explicitly prioritized in the model. This can be reached by giving different weights on every artery in the network. This discrimination of arteries will produce a decrease in the efficiency of the use of traffic lights in those less important in favour of those weighted more highly. Therefore, the calibration of these parameters must be careful and based on traffic volume statistics.

Let A be the set of arteries on the network. We define k_a and $\overline{k}_a$ as the weights for the outbound and inbound bandwidth at artery a, respectively. The objective function to be maximized is then:

$$\sum_{a\in A} \left(k_a b_a + \overline{k}_a \overline{b}_a\right).$$

2.3.2 Arterial Constraints

As said before, we assume that all signals work into a common signal period whose length is introduced in the model as a decision variable. It must lie in an interval $[T_1, T_2]$, see (2.2). Decision variable z is the reciprocal of the period length, that is, $z = 1/T$. Inequalities (2.3)-(2.4) ensure that the bandwidth remains within the green time. The velocities v_i^a between each signal on each artery are decision variables and bounded, with e_i^a and f_i^a representing the lower and upper limits, respectively, and d_i^a is the distance between two consecutive arteries, see (2.6)-(2.7). In order to avoid sudden changes in the velocities between consecutive signals, they are limited by imposing lower and upper bounds $1/h_i^a$ and $1/g_i^a$ on changes in reciprocal velocities. The reason for using reciprocal bounds for velocities and changes in velocities is that linear constraints can be obtained in this way. It is not possible to consider directly the inequalities $e_i^a \le v_i^a \le f_i^a$ because the period length is also a variable. Thus, if we try to pass from v_i^a in meters/second to v_i^a in meters/period, we will have $e_i^a T \le v_i^a T \le f_i^a T$, which are nonlinear constraints. Therefore, we use the reciprocals of e_i^a and f_i^a,

$$e_i^a \le v_i^a \le f_i^a \quad \rightarrow \quad \frac{d_i^a}{f_i^a} \le \frac{d_i^a}{v_i^a} \le \frac{d_i^a}{e_i^a} \quad \rightarrow \quad \frac{d_i^a}{f_i^a} z \le t_i^a \le \frac{d_i^a}{e_i^a} z.$$

The same applies to the changes of velocities.

It can be seen in Figure 2.4 that $Time_{\mathsf{A\text{-}B}} = \Delta_{ai}$+integer number of periods+ϕ_{ij}^a and that $Time_{\mathsf{A\text{-}B}}$ =integer number of periods$-\overline{\phi}_{ij}^a$+integer number of periods+Δ_{aj}. Therefore:

$$\phi_{ij}^a + \overline{\phi}_{ij}^a + \Delta_{ai} - \Delta_{aj} = m_{ij}^a, \tag{2.15}$$

where m_{ij}^a is an integer decision variable (number of periods). Also, $Time_{\mathsf{C\text{-}D}} = \phi_{ij}^a + \frac{1}{2} r_{aj} + w_{aj} + \tau_{aj} = \frac{1}{2} r_{ai} + w_{ai} + t_{ij}^a$ and $Time_{\mathsf{E\text{-}F}} = \overline{\phi}_{ij}^a + \frac{1}{2}\overline{r}_{aj} + \overline{w}_{aj} = \frac{1}{2}\overline{r}_{ai} + \overline{w}_{ai} - \overline{\tau}_{ai} + \overline{t}_{ij}^a$. So, if we now substitute in (2.15), we have that

$$\begin{aligned} t_{ij}^a + \overline{t}_{ij}^a + \frac{1}{2}(r_{ai} + \overline{r}_{ai}) + (w_{ai} + \overline{w}_{ai}) - \frac{1}{2}(r_{aj} + \overline{r}_{aj}) - (w_{aj} + \overline{w}_{aj}) - (\tau_{aj} + \overline{\tau}_{ai}) \\ + (\Delta_{ai} - \Delta_{aj}) = m_{ij}^a. \end{aligned} \tag{2.16}$$

Equation (2.16) is named *arterial loop constraint* for artery a between signals S_{ai} and S_{aj}.

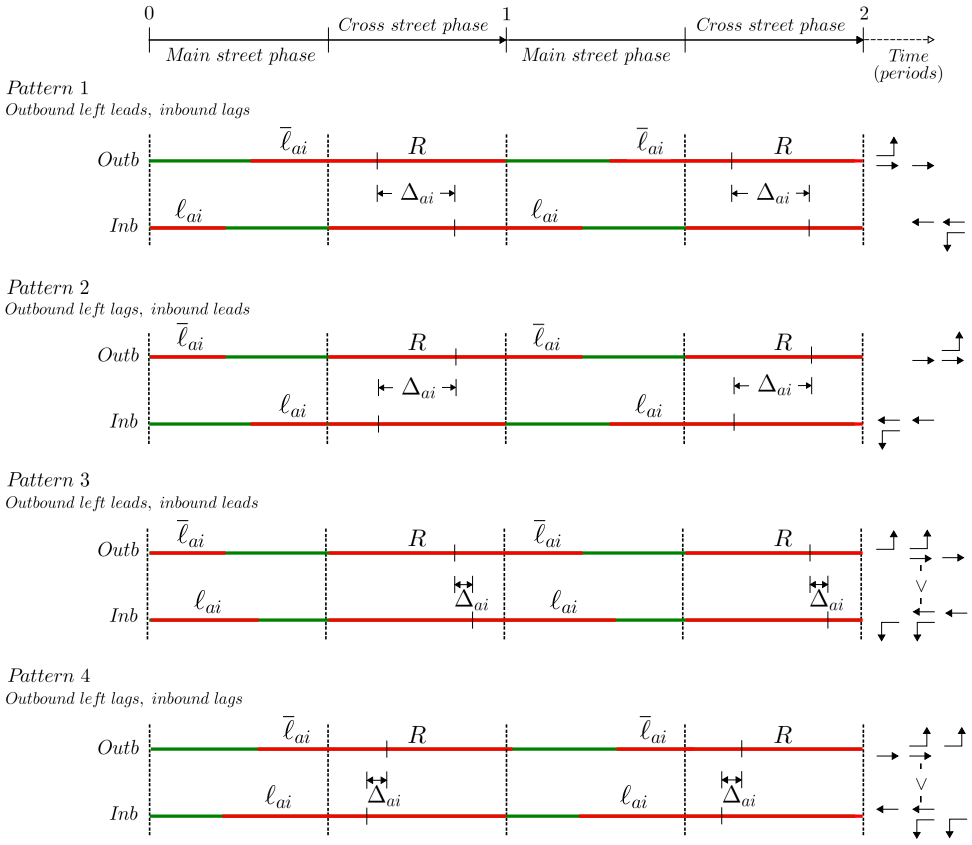


Figure 2.6: Patterns of left turn phases.

In addition, there are constraints that model left turn decisions if it is allowed by green lights. The MAXBAND allows to decide among four possible patterns of left turns which can be seen in Figure 2.6 (see Little et al. (1981) for more details).

Parameters ℓ_{ai} and $\overline{\ell}_{ai}$ in Figure 2.6 represent, for a signal i on an artery a, the time (periods) of outbound and inbound left turn phases respectively. R is the common red time. For instance, Figure 2.7 shows the three possible movements for vehicles on a main street in three different moments. As can be seen at area 2, the traffic lights are green for outbound and inbound directions, so no car in the horizontal street can cross to the other side. On the common red time R a possible different left turn pattern can be given for cross street.

Furthermore, Δ_{ai} can be expressed as a function of ℓ_{ai} and $\overline{\ell}_{ai}$. For example, if we consider Pattern 1 and we calculate the difference between the center of total red time of outbound and the total red of inbound (in that order), we have that

$$\Delta_{ai} = \frac{\overline{\ell}_{ai} + R}{2} - \left(\frac{R + \ell_{ai}}{2} + \overline{\ell}_{ai} \right) = -\frac{\ell_{ai} + \overline{\ell}_{ai}}{2}.$$

The results for the other left turn phases are shown in Table 2.1. All these expressions can be obtained with the formula:

$$\Delta_{ai} = \frac{1}{2}[(2\delta_{ai} - 1)\ell_{ai} - (2\overline{\delta}_{ai} - 1)\overline{\ell}_{ai}], \tag{2.17}$$

where δ_{ai}, $\overline{\delta}_{ai} \in \{0, 1\}$ are additional binary variables. The decisions on left turns are included in the model by substituting equation (2.17) in (2.16), as can be seen in constraint (2.5).

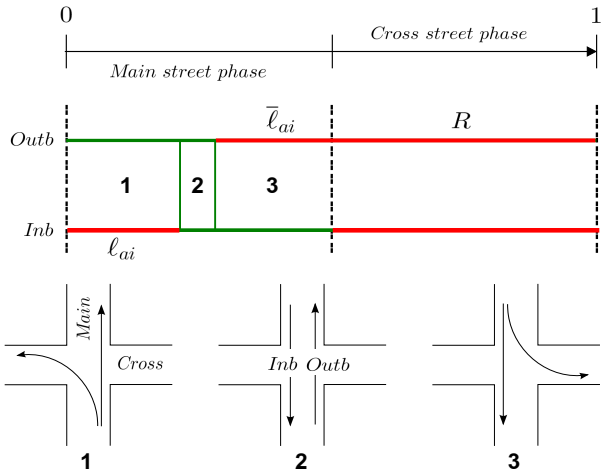


Figure 2.7: Left turn phase example with Pattern 1.

Table 2.1: Expressions for Δ_{ai}'s.

Pattern	Δ_{ai}	δ_{ai}	$\overline{\delta}_{ai}$
1	$-\frac{\ell_{ai}+\overline{\ell}_{ai}}{2}$	0	1
2	$\frac{\ell_{ai}+\overline{\ell}_{ai}}{2}$	1	0
3	$-\frac{\ell_{ai}-\overline{\ell}_{ai}}{2}$	0	0
4	$\frac{\ell_{ai}-\overline{\ell}_{ai}}{2}$	1	1

2.3.3 Loop Constraints

The network case is a natural generalization of the arterial case and the corresponding model includes all the constraints for each artery as shown before. The arterial loop constraints (2.16) can be seen as a cycle for two nodes because it represents the movement of going to and returning from a signal. If we extend this idea to larger cycles, it is clear that the sum of all the offsets in the cycle must be an integer number as well.

In order to see how to write the equation of the cycle constraints, let us start with an example. A cycle with 4 arteries $A = \{a, b, c, d\}$ and 4 junctions $J = \{J_1, J_2, J_3, J_4\}$ is shown in Figure 2.8. Each artery $a \in A$ has signals S_{aj}, where j is the index for signals on a increasing in the outbound direction given by the arrows. Heading in the clockwise direction and starting from junction J_1, the cycle constraint for this example is:

$$\phi^b_{jk} + \Psi^{J_2}_{S_{bk},S_{co}} + \phi^c_{op} + \Psi^{J_3}_{S_{cp},S_{dq}} + \phi^d_{qr} + \Psi^{J_4}_{S_{dr},S_{ah}} + \phi^a_{hi} + \Psi^{J_1}_{S_{ai},S_{bj}} = C_\zeta,$$

where C_ζ is an integer decision variable and $\Psi^i_{S_{aj},S_{bk}}$ is a decision variable named *intranode offset* which represents the time between consecutive centers of reds for signals S_{aj} and S_{bk} that meet at junction i, i.e, it is a link time between arteries a and b.

In order to generalize the previous expression to any cycle, we define the following sets:

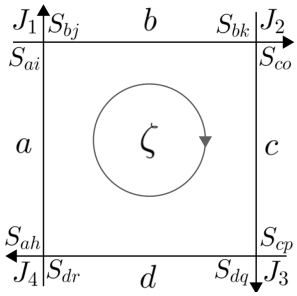


Figure 2.8: Clockwise loop with 4 junctions.

- A_ζ^F (A_ζ^B): Set of all segments of forward (backward) arteries with edges (i, j) in the clockwise direction of cycle ζ,
- J_ζ: All sets of the form (b, j, i, c, k) in ζ, where i is the junction between arteries b and c in the signals S_{bj} and S_{ck},

Then, the *network loop constraint* (cycle constraint) is:

$$\sum_{(i,j):a\in A_\zeta^F} \phi_{ij}^a \; - \sum_{(i,j):a\in A_\zeta^B} \phi_{ij}^a \; + \sum_{(b,j,i,c,k)\in J_\zeta} \Psi^i_{S_{bj},S_{ck}} = C_\zeta.$$

The number of cycle constraints in the model will depend on how many edges and nodes the network has. Unfortunately, this number can be very high, which makes the problem very difficult to solve. The following result helps us to alleviate this problem: It is well known that the set of all cycles ζ on any single graph can be spanned by a basis $\mathcal{B}_\zeta$ with cardinality $m-n+1$, where m represents the number of edges and n the number of nodes on the underlying undirected graph related to the directed graph that represents the original network. See Liebchen and Rizzi (2007) for full details. So, a cycle basis must be found before writing down the model.

2.3.4 Computing Intranode Offset

In general, a grid graph $G_{k\times k}$ needs $(k-1)^2$ network loop constraints and therefore several intranodes must be computed for each of these equations. The values of intranode offsets $\Psi^i_{S_{aj},S_{bk}}$'s depend on red times positions of signals S_{aj} and S_{bk} on a main and cross street, respectively. For instance, if left turns are not permitted, then the red and green times for main and cross street will have the same length and in this case the intranode offset is clearly 0.5 periods. Since the MAXBAND model must decide among four different left turn patterns on each junction i, the computation of each intranode should take into account all the possible values of binary variables involved in that choice. The next result provides a simple expression to compute intranode offsets which does not require of extra variables beyond those used in model LM 2.3.1.

Theorem 2. *Consider the patterns of left turn phases shown in Figure 2.6. Let $\psi_{mc}^{p_m p_c}$ be the value of $\Psi^i_{S_{mj},S_{ck}}$ when arteries m and c meet at junction i for signals S_{mj} and S_{ck} with left turn phases patterns p_m and p_c respectively. Then, for all possible values of p_m and p_c, we have that:*

$$\Psi^i_{S_{mj},S_{ck}} = \frac{1}{2} - \frac{1}{2}\left[(2\overline{\delta}_{ck}-1)\overline{\ell}_{ck} - (2\overline{\delta}_{mj}-1)\overline{\ell}_{mj}\right].$$

Proof. Let us consider Figure 2.9.

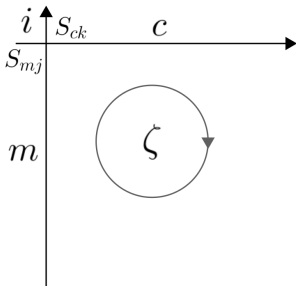


Figure 2.9: A junction i of arteries main (m) and cross (c).

Figure 2.10 shows the different forms that $\psi_{mc}^{p_m p_c}$ may have for all possible permutations of left turn phases in Figure 2.6. The cross street phase takes place during the red time R_m in the main street and vice versa.

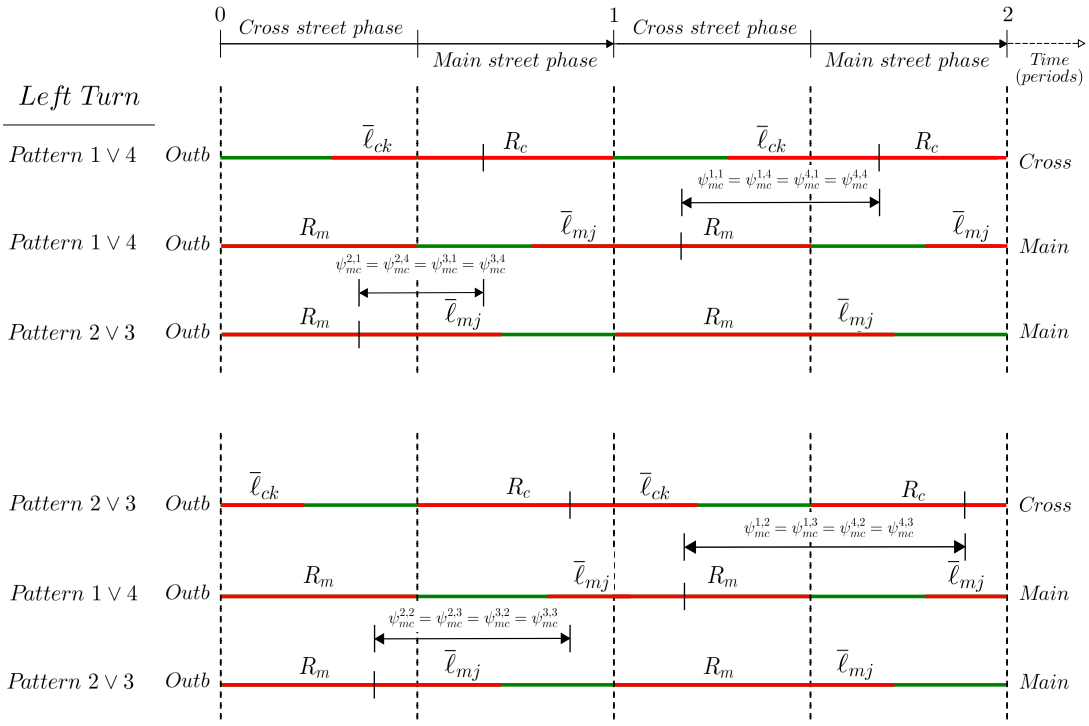


Figure 2.10: Geometry for $\psi_{mc}^{p_m p_c}$.

All values of $\psi_{mc}^{p_m p_c}$ are summarized in Table 2.2. We take into account only the outbound directions phases because we are using only the ϕ's in equation (2.10) of the MAXBAND model.

Also, in Table 2.2 it can be seen that:

$$\psi_{mc}^{1,1} = \psi_{mc}^{1,4} = \psi_{mc}^{4,1} = \psi_{mc}^{4,4} = \frac{1-\overline{\ell}_{ck}+\overline{\ell}_{mj}}{2},\ \psi_{mc}^{2,1} = \psi_{mc}^{2,4} = \psi_{mc}^{3,1} = \psi_{mc}^{3,4} = \frac{1-\overline{\ell}_{ck}-\overline{\ell}_{mj}}{2},$$

$$\psi_{mc}^{1,2} = \psi_{mc}^{1,3} = \psi_{mc}^{4,2} = \psi_{mc}^{4,3} = \frac{1+\overline{\ell}_{ck}+\overline{\ell}_{mj}}{2}, \text{ and } \psi_{mc}^{2,2} = \psi_{mc}^{2,3} = \psi_{mc}^{3,2} = \psi_{mc}^{3,3} = \frac{1+\overline{\ell}_{ck}-\overline{\ell}_{mj}}{2}.$$

In Table 2.3 all different values of the binary variables δ's and $\overline{\delta}$'s are shown for each left turn phase that the model uses to compute the Δ's with equation (2.17). The $\psi_{mc}^{p_m p_c}$'s are arranged in four groups determined just for the values of $\overline{\delta}$'s on the signals S_{mj} and S_{ck}.

Table 2.2: Expressions for $\psi_{mc}^{p_m p_c}$'s.

		Cross Street			
	Patterns	1	2	3	4
Main Street	1	$\frac{1-\overline{\ell}_{ck}+\overline{\ell}_{mj}}{2}$	$\frac{1+\overline{\ell}_{ck}+\overline{\ell}_{mj}}{2}$	$\frac{1+\overline{\ell}_{ck}+\overline{\ell}_{mj}}{2}$	$\frac{1-\overline{\ell}_{ck}+\overline{\ell}_{mj}}{2}$
	2	$\frac{1-\overline{\ell}_{ck}-\overline{\ell}_{mj}}{2}$	$\frac{1+\overline{\ell}_{ck}-\overline{\ell}_{mj}}{2}$	$\frac{1+\overline{\ell}_{ck}-\overline{\ell}_{mj}}{2}$	$\frac{1-\overline{\ell}_{ck}-\overline{\ell}_{mj}}{2}$
	3	$\frac{1-\overline{\ell}_{ck}-\overline{\ell}_{mj}}{2}$	$\frac{1+\overline{\ell}_{ck}-\overline{\ell}_{mj}}{2}$	$\frac{1+\overline{\ell}_{ck}-\overline{\ell}_{mj}}{2}$	$\frac{1-\overline{\ell}_{ck}-\overline{\ell}_{mj}}{2}$
	4	$\frac{1-\overline{\ell}_{ck}+\overline{\ell}_{mj}}{2}$	$\frac{1+\overline{\ell}_{ck}+\overline{\ell}_{mj}}{2}$	$\frac{1+\overline{\ell}_{ck}+\overline{\ell}_{mj}}{2}$	$\frac{1-\overline{\ell}_{ck}+\overline{\ell}_{mj}}{2}$

Table 2.3: Four different groups of $\psi_{mc}^{p_m p_c}$'s on junction i.

Patterns						
p_m	p_c	δ_{mj}	$\overline{\delta}_{mj}$	δ_{ck}	$\overline{\delta}_{ck}$	$\psi_{mc}^{p_m p_c}$
4	4	1	1	1	1	$\psi_{mc}^{4,4}$
4	1	1	1	0	1	$\psi_{mc}^{4,1}$
1	4	0	1	1	1	$\psi_{mc}^{1,4}$
1	1	0	1	0	1	$\psi_{mc}^{1,1}$
3	4	0	0	1	1	$\psi_{mc}^{3,4}$
2	1	1	0	0	1	$\psi_{mc}^{2,1}$
3	1	0	0	0	1	$\psi_{mc}^{3,1}$
2	4	1	0	1	1	$\psi_{mc}^{2,4}$
1	2	0	1	1	0	$\psi_{mc}^{1,2}$
4	2	1	1	1	0	$\psi_{mc}^{4,2}$
4	3	1	1	0	0	$\psi_{mc}^{4,3}$
1	3	0	1	0	0	$\psi_{mc}^{1,3}$
2	2	1	0	1	0	$\psi_{mc}^{2,2}$
2	3	1	0	0	0	$\psi_{mc}^{2,3}$
3	2	0	0	1	0	$\psi_{mc}^{3,2}$
3	3	0	0	0	0	$\psi_{mc}^{3,3}$

It is now easy to verify that a single expression to compute any $\Psi^i_{S_{mj},S_{ck}}$ is given by:

$$\Psi^i_{S_{mj},S_{ck}} = \frac{1}{2} - \frac{1}{2}\left[(2\overline{\delta}_{ck}-1)\overline{\ell}_{ck} - (2\overline{\delta}_{mj}-1)\overline{\ell}_{mj}\right]. \tag{2.18}$$

□

A similar result that considers pedestrian crossing times can be found in Chaudhary (1987). However it must be noted the result that is shown here was developed independently.

2.4 Bounds for Integer Variables

Little (1966) provides bounds for the integer variables in the arterial loop constraints when left turns phases are not included. In this section we extend those bounds for the integer variables

in the network loop constraints.

It is clear from Figure 2.4 that $0 \leq w_{ai} \leq 1 - r_{ai}$ and that $0 \leq \overline{w}_{ai} \leq 1 - \overline{r}_{ai}$ for any signal i on artery a. By using constraints (2.2), (2.6) and (2.7) we have that $\frac{d^a_{ij}}{f^a_{ij}T_2} \leq t^a_{ij} \leq \frac{d^a_{ij}}{e^a_{ij}T_1}$ and that $\frac{d^a_{ij}}{\overline{f}^a_{ij}T_2} \leq \overline{t}^a_{ij} \leq \frac{d^a_{ij}}{\overline{e}^a_{ij}T_1}$. Furthermore, equation (2.17) gives $\Delta_{ai} = \delta_{ai}\ell_{ai} - \frac{\ell_{ai}}{2} - \overline{\delta}_{ai}\overline{\ell}_{ai} + \frac{\overline{\ell}_{ai}}{2}$. Therefore, we have that $-(\frac{\ell_{ai}}{2} + \frac{\overline{\ell}_{ai}}{2}) \leq \Delta_{ai} \leq (\frac{\ell_{ai}}{2} + \frac{\overline{\ell}_{ai}}{2})$, as δ's take values in $\{0, 1\}$.

If we now use these bounds in constraints (2.5), then we have the following limits for m's in terms of the MAXBAND model's parameters, $\underline{m^a_{ij}} \leq m^a_{ij} \leq \overline{m^a_{ij}}$, where:

$$\overline{m^a_{ij}} = \left\lfloor 2 - \frac{1}{2}(r_{ai} + \overline{r}_{ai}) - \frac{1}{2}(r_{aj} + \overline{r}_{aj}) + \frac{1}{2}(\ell_{ai} + \overline{\ell}_{ai}) + \frac{1}{2}(\ell_{aj} + \overline{\ell}_{aj}) - (\tau_{aj} + \overline{\tau}_{ai}) + \frac{d^a_{ij}}{e^a_{ij}T_1} + \frac{d^a_{ij}}{\overline{e}^a_{ij}T_1} \right\rfloor, \tag{2.19}$$

$$\underline{m^a_{ij}} = \left\lceil -2 + \frac{1}{2}(r_{ai} + \overline{r}_{ai}) + \frac{1}{2}(r_{aj} + \overline{r}_{aj}) - \frac{1}{2}(\ell_{ai} + \overline{\ell}_{ai}) - \frac{1}{2}(\ell_{aj} + \overline{\ell}_{aj}) - (\tau_{aj} + \overline{\tau}_{ai}) + \frac{d^a_{ij}}{f^a_{ij}T_2} + \frac{d^a_{ij}}{\overline{f}^a_{ij}T_2} \right\rceil. \tag{2.20}$$

Therefore, following the notation in LM 2.3.1, i.e., $m^a_{i,i+1} = m^a_i$, we have the following bounds:

$$\underline{m^a_i} \leq m^a_i \leq \overline{m^a_i}, \quad \forall a \in A, \forall i = 1, \ldots, n_a - 1. \tag{2.21}$$

Furthermore, let $a \in A^F_\zeta$ be, as in previous sections, an artery with signals S_{ai}, where $i \in I^a_\zeta = \{1^a_\zeta, \ldots, n^a_\zeta\}$ and increasing in the outbound direction. The time between the first signal and the last one in the artery segment is:

$$t^a_{1^a_\zeta, n^a_\zeta} = \sum_{i \in I^a_\zeta \setminus \{n^a_\zeta\}} t^a_i - \sum_{i \in I^a_\zeta \setminus \{1^a_\zeta\}} \tau_{ai}. \tag{2.22}$$

Therefore, $\underline{t^a_{1^a_\zeta, n^a_\zeta}} \leq t^a_{1^a_\zeta, n^a_\zeta} \leq \overline{t^a_{1^a_\zeta, n^a_\zeta}}$, where:

$$\overline{t^a_{1^a_\zeta, n^a_\zeta}} = \sum_{i \in I^a_\zeta \setminus \{n^a_\zeta\}} \frac{d^a_i}{e^a_i T_1} - \sum_{i \in I^a_\zeta \setminus \{1^a_\zeta\}} \tau_{ai}, \tag{2.23}$$

$$\underline{t^a_{1^a_\zeta, n^a_\zeta}} = \sum_{i \in I^a_\zeta \setminus \{n^a_\zeta\}} \frac{d^a_i}{f^a_i T_2} - \sum_{i \in I^a_\zeta \setminus \{1^a_\zeta\}} \tau_{ai}. \tag{2.24}$$

By using the same facts mentioned above and since $\phi^a_{ij} + \frac{1}{2}r_{aj} + w_{aj} + \tau_{aj} = \frac{1}{2}r_{ai} + w_{ai} + t^a_{ij}$, we obtain that $\underline{\phi^a_{i^a_\zeta, n^a_\zeta}} \leq \phi^a_{i^a_\zeta, n^a_\zeta} \leq \overline{\phi^a_{i^a_\zeta, n^a_\zeta}}$, with:

$$\overline{\phi^a_{i^a_\zeta, n^a_\zeta}} = -\frac{1}{2}(r_{a,1^a_\zeta} + r_{a,n^a_\zeta}) + \overline{t^a_{1^a_\zeta, n^a_\zeta}} + 1, \tag{2.25}$$

$$\underline{\phi^a_{i^a_\zeta, n^a_\zeta}} = \frac{1}{2}(r_{a,1^a_\zeta} + r_{a,n^a_\zeta}) + \underline{t^a_{1^a_\zeta, n^a_\zeta}} - 1. \tag{2.26}$$

According to equation (2.18), we also have that $\underline{\Psi^i_{S_{bj}, S_{ck}}} \leq \Psi^i_{S_{bj}, S_{ck}} \leq \overline{\Psi^i_{S_{bj}, S_{ck}}}$:

$$\overline{\Psi^i_{S_{bj},S_{ck}}} = \frac{1}{2}(1 + \overline{\ell}_{ck} + \overline{\ell}_{bj}), \tag{2.27}$$

$$\underline{\Psi^i_{S_{bj},S_{ck}}} = \frac{1}{2}(1 - \overline{\ell}_{ck} - \overline{\ell}_{bj}). \tag{2.28}$$

Finally, bounds for C_ζ in the set of constrains (2.10) are:

$$\overline{C_\zeta} = \left\lfloor \sum_{a\in A^F_\zeta} \overline{\phi^a_{i^a_\zeta,n^a_\zeta}} - \sum_{a\in A^B_\zeta} \underline{\phi^a_{i^a_\zeta,n^a_\zeta}} + \sum_{(b,j,i,c,j)\in J_\zeta} \overline{\Psi^i_{S_{bj},S_{ck}}} \right\rfloor, \tag{2.29}$$

$$\underline{C_\zeta} = \left\lceil \sum_{a\in A^F_\zeta} \underline{\phi^a_{i^a_\zeta,n^a_\zeta}} - \sum_{a\in A^B_\zeta} \overline{\phi^a_{i^a_\zeta,n^a_\zeta}} + \sum_{(b,j,i,c,j)\in J_\zeta} \underline{\Psi^i_{S_{bj},S_{ck}}} \right\rceil. \tag{2.30}$$

The next set of constraints can be added to the linear model:

$$\underline{C_\zeta} \leq C_\zeta \leq \overline{C_\zeta}, \quad \forall \zeta \in \mathcal{B}_\zeta. \tag{2.31}$$

All these limits will be used for our computational experiments in the next section, in order to reduce the space of search of values of the integer variables.

2.5 The First Systematic Algorithm for the Arterial Case

In this section we review a classic study on bandwidth maximization on an arterial road proposed by Morgan and Little (1964) in 1964. They present a geometric and intuitive method to solve this case, very well supported by a sequence of theoretical results. Although there have been geometric methods developed earlier, this work was the first to provide a systematic algorithm that could be implemented computationally.

Most of the proofs have been omitted but we have kept those that we consider important for the understanding of the complete procedure.

2.5.1 The Theory

The method presented in Morgan and Little (1964) solves the next two problems:

- **Problem 1.** Given an arbitrary number of signals along a street, a common signal period, the green and red times for each signal, and specified vehicle speeds in each direction between adjacent signals, synchronize the signals to produce bandwidths that are equal in each direction and as large as possible.
- **Problem 2.** Resynchronize to favour one direction with a larger bandwidth, if feasible, and give the other direction the largest bandwidth possible.

It is assumed that the signals have a common period T, i.e., if r_i and g_i are the red time and the green time of signal i, respectively, then $r_i + g_i = r_j + g_j$ for all signals i and j. Furthermore, it is considered $\Delta_i = w_i = 0$. Therefore, and unlike Figure 2.4, the red and green times overlap each other in each signal for both directions, as we can see in Figure 2.11. In order to ease the notation we have omitted the name of the artery a in each of the definitions.

Additionally, let:

- θ_{ij}: Time from the center of a red time of S_i to the next center of red of S_j. It is called relative phase or offset. By convention $0 \leq \theta_{ij} < 1$, in periods.
- x_i: Position of S_i on the street, in distance units.
- $v_k(\overline{v}_k)$: Outbound (inbound) speed between signal S_k and S_{k+1}. It is considered fixed and known for all $k \in \{1, \ldots, n-1\}$, in distance per time.

Furthermore, Morgan and Little (1964) defined $\overline{t}_{ij}$ as the travel time from S_i to S_j in the inbound direction and in this case the values de $\overline{t}_{ij}$ can take a negative sign. If $j = i + 1$, $\overline{t}_{ij}$ must be computed as: $t_{i,i+1} = \dfrac{x_{i+1} - x_i}{v_i T}$ and $\overline{t}_{i,i+1} = \dfrac{x_i - x_{i+1}}{\overline{v}_i T}$.

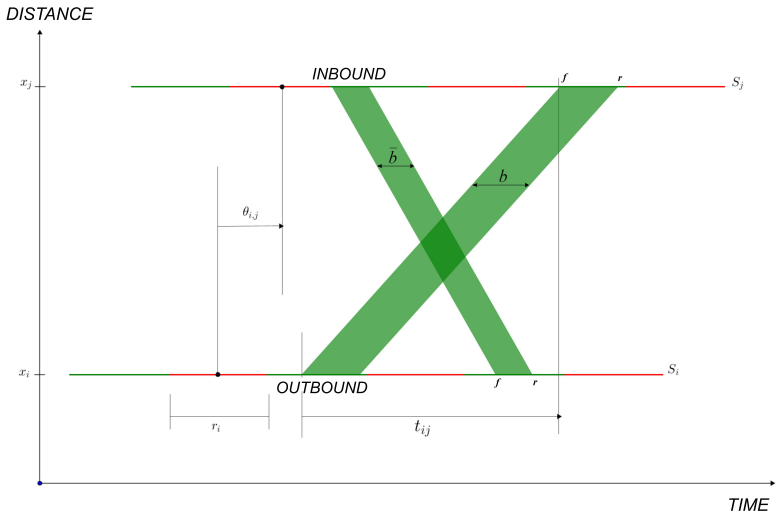


Figure 2.11: Distance-Time diagram.

Definition 3 (Synchronization, Morgan and Little (1964)). *It is a set* $\{\theta_{ij} \mid j = 1, \ldots, n\}$ *for* $i \in \{1, \ldots, n\}$.

As mentioned before, the method is based on a consistent sequence of theorems.

Definition 4 (Critical Signal, Morgan and Little (1964)). *A signal* S_i *is said to be a critical signal if one side of* S_i*'s red touches the green band in one direction and the other side touches the green band in the other direction.*

Lemma 1 (Morgan and Little (1964)). *If a synchronization maximizes* $b + \overline{b}$ *subject to* $b > 0$ *and* $\overline{b} > 0$*, then:*

1. *There exists at least one critical signal.*
2. *The red time of any critical signal will touch the front edge of one green band and the rear edge of the other.*
3. *All critical signals can be divided into two groups:*
   - *Group 1: consists of signals whose reds touch the front of outbound and the rear of inbound (see Figure 2.12), and*

- *Group 2: of signals whose reds touch the front of inbound and the rear of outbound (see Figure 2.13).*

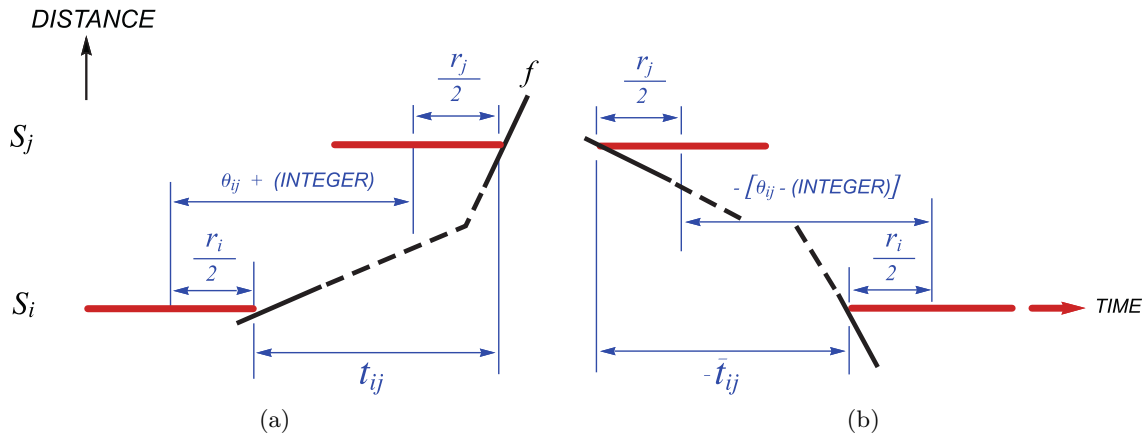


Figure 2.12: Geometry for two signals in the group 1.

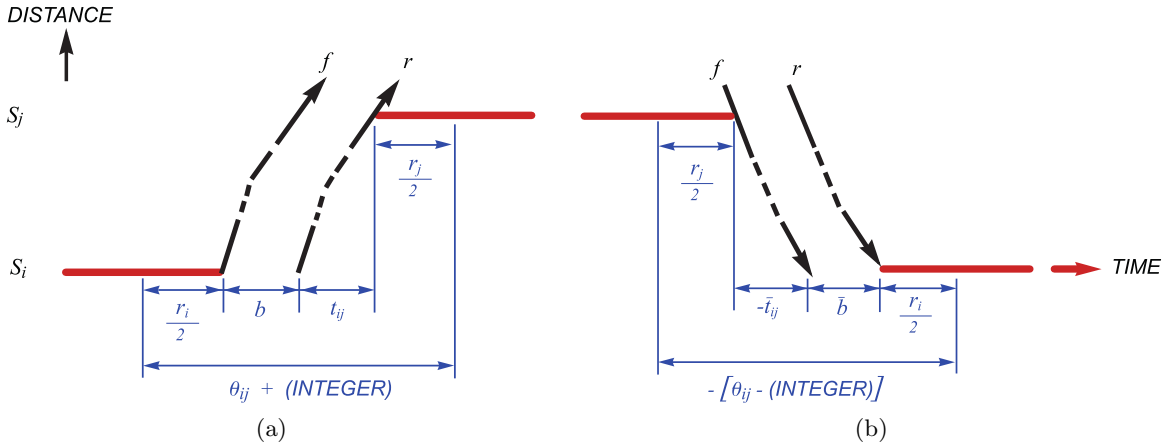


Figure 2.13: Geometry for two signals in different groups.

From Figure 2.12 we can note that $\frac{1}{2}r_i + t_{ij} = \frac{1}{2}r_j + \theta_{ij} + (int)$ and $\frac{1}{2}r_i - \bar{t}_{ij} = \frac{1}{2}r_j - \theta_{ij} + (int)$, where $int = integer$ and represents an integer that is added to keep θ within the range $[0, 1)$. Note that depending on the speed, int will increase as much as periods have passed. Now, if we add both expressions, it gives:

$$\theta_{ij} = \frac{1}{2}(t_{ij} + \bar{t}_{ij}) + \frac{1}{2}(int) \tag{2.32}$$

If we consider a scheme with two signals in group 2 we obtain the same equation.

Suppose that $0 \leq \theta_{ij} < 1$. Then a more explicit expression can be obtained by using the function mantissa (man): $man(\#) = \# - floor(\#)$, where $\# \in \mathbb{R}$.

Then, from Equation (2.32):

Definition 5 (Half-Integer Synchronization, Morgan and Little (1964))**.** *A Phasing or Half-Integer Synchronization is*

$$\theta_{ij} = man\left[\frac{1}{2}(t_{ij} + \bar{t}_{ij}) + \delta_{ij}\right] \tag{2.33}$$

where $\delta_{ij} \in \{0, \frac{1}{2}\}$.

With the above arguments we have proved the next lemma:

Lemma 2 (Morgan and Little (1964)). *Under the conditions of Lemma 1, each group of signals has half-integer synchronization.*

Theorem 3 (Morgan and Little (1964)). *There is a half-integer synchronization that gives maximal equal bandwidths.*

The theorem implies that if the maximal equal bandwidths are greater than zero, $max(b+\overline{b})$ s.t. $b, \overline{b} > 0$ equals $max(b+\overline{b})$ s.t. $b = \overline{b}$. The scheme of the proof in Morgan and Little (1964) is basically a geometric construction of the half-integer synchronization and easy to follow. The proof also establishes that there is a half-integer synchronization between two signals from different groups. From Figure 2.13 we have that $\frac{1}{2}r_i + b + t_{ij} + \frac{1}{2}r_j = \theta_{ij} + (int)$ and that $\frac{1}{2}r_i + \overline{b} - \overline{t}_{ij} + \frac{1}{2}r_j = -\theta_{ij} + (int)$, their difference yields $b - \overline{b} + t_{ij} + \overline{t}_{ij} = 2\theta_{ij} + (int)$ and since it is possible to obtain $b = \overline{b}$ then $\theta_{ij} = \frac{1}{2}(t_{ij} + \overline{t}_{ij}) + \frac{1}{2}(int)$. Furthermore, if in the difference we replace $\theta_{ij} = \frac{1}{2}(t_{ij} + \overline{t}_{ij}) + \frac{1}{2}(int)$, then it will give $b = \overline{b}$. Therefore:

Theorem 4 (Morgan and Little (1964)). *Under any half-integer synchronization,* $b = \overline{b}$.

2.5.2 The Algorithm

Due to Theorems 3 and 4 just a few cases need to be considered in order to develop an algorithm to solve Problems 1 and 2. By Theorem 3 it is enough to examine Half-Integer Synchronizations and by Theorem 4 it suffices to focus only on the outbound direction.

Consider u_{ij} just as in Figure 2.14:

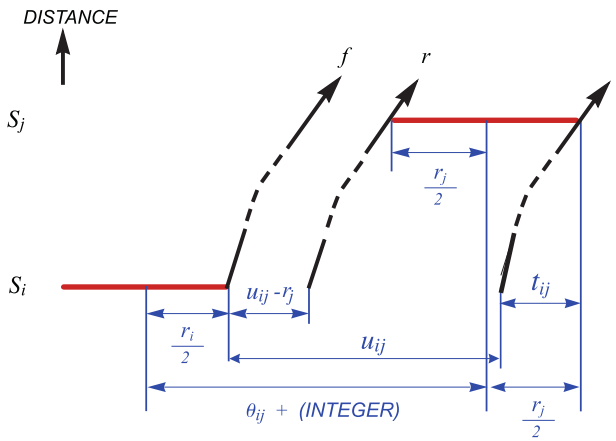


Figure 2.14: Geometry for SEB Procedure.

Theorem 5 (Morgan and Little (1964)). *The maximal equal bandwidth is* $max\{0, B\}$, *where*

$$B = max_i min_j max_{\delta \in \{0,1/2\}} \{u_{ij}(\delta) - r_j\}.$$

Let $i = c$ *be a maximizing* i *and* $\delta_{c1}, \ldots, \delta_{cn}$ *be the corresponding maximizing* δ*'s. Then, a synchronization for maximal equal bandwidths is* $\theta_{c1}, \ldots, \theta_{cn}$ *obtained by substituting the* δ_{cj} *into* $\theta_{ij} = man\left[\frac{1}{2}(t_{ij} + \overline{t}_{ij}) + \delta_{ij}\right]$

Proof. Referring to Figure 2.14, we have $u_{ij} = man[\theta_{ij} + \frac{r_j}{2} - \frac{r_i}{2} - t_{ij}]$, but in order to make $u_{ij} = 1$ when this expression is zero, it can be written $u_{ij} = 1 - man[-\theta_{ij} - \frac{r_j}{2} + \frac{r_i}{2} + t_{ij}]$ and substituting θ from the equation 2.33, we have,

$$u_{ij}(\delta_{ij}) = 1 - man\left[\frac{1}{2}(r_i - r_j) + \frac{1}{2}(t_{ij} - \overline{t}_{ij}) - \delta_{ij}\right]. \tag{2.34}$$

Due to $\delta_{ij} \in \{0, \frac{1}{2}\}$ and Figure 2.14, the best value for δ_{ij} can be reached by

$$max_{\delta\in\{0,1/2\}}\left[u_{ij}(\delta) - r_j\right].$$

Let

- b_i : Greatest outbound bandwidth under half-integer synchronization if S_i's red touches the front of the outbound band.
- B : The value of one of the maximal equal bandwidths.

Then, $b_i = min_j max_{\delta\in\{0,1/2\}}\left[u_{ij}(\delta) - r_j\right]$ because the trajectories should not cross the red lines. Therefore the best i is such that

$$B = max_i min_j max_{\delta\in\{0,1/2\}}\{u_{ij}(\delta) - r_j\}$$

If the best i is c, and $\delta_{c1}, \ldots, \delta_{cn}$ are the maximizing δ's, then the synchronization will be reached by substituting δ_{cj}'s in the Equation (2.33) to obtain the set $\{\theta_{c1}, \ldots, \theta_{cn}\}$

□

The theorem above is the base for the SEB Procedure (Synchronization for Maximal Equal Bandwidths) and it is summarized in Figure 2.15.

Clearly a platoon (group of cars) needs some time to cross a traffic light, and the length of the platoon (measured in seconds) could be different in both directions. The authors end their analysis with this fact being considered. The crossing time of the platoon affects the synchronization, so the first step is to evaluate how far a red line must be moved in order to avoid the cars stopping on the corners. It is clear that there is a limit to that movement.

Theorem 6 (The Shifting Procedure, Morgan and Little (1964))**.** *Let g be the smallest green time. The outbound bandwidth b can be assigned any value in $max\{0, B\} \leq b \leq g$ by making a phase shift*

$$\alpha_j = max\{u_{cj} - 1 + b - B, 0\}.$$

Then $\overline{b} = max\{2B - b, 0\}$ and $\overline{b}$ is as large as possible for a given b.
Alternatively, the inbound bandwidth $\overline{b}$, can be assigned any value in $max\{0, B\} \leq \overline{b} \leq g$, by making a phase shift

$$\alpha_j = max\{\overline{b} + r_j - u_{cj}, 0\}.$$

Then $b = max\{2B - \overline{b}, 0\}$ and b is as large as possible for a given $\overline{b}$.

Proof. Let us define:

- $\theta_{c1}, \ldots, \theta_{cn}$: The maximal equal bandwidth reached with the SEB Procedure with S_c the critical signal whose red touches the front of the outbound green band (see Figure 2.16). The corresponding $u_{c1}, \ldots, u_{cn}$ and B are supposed to be known too.

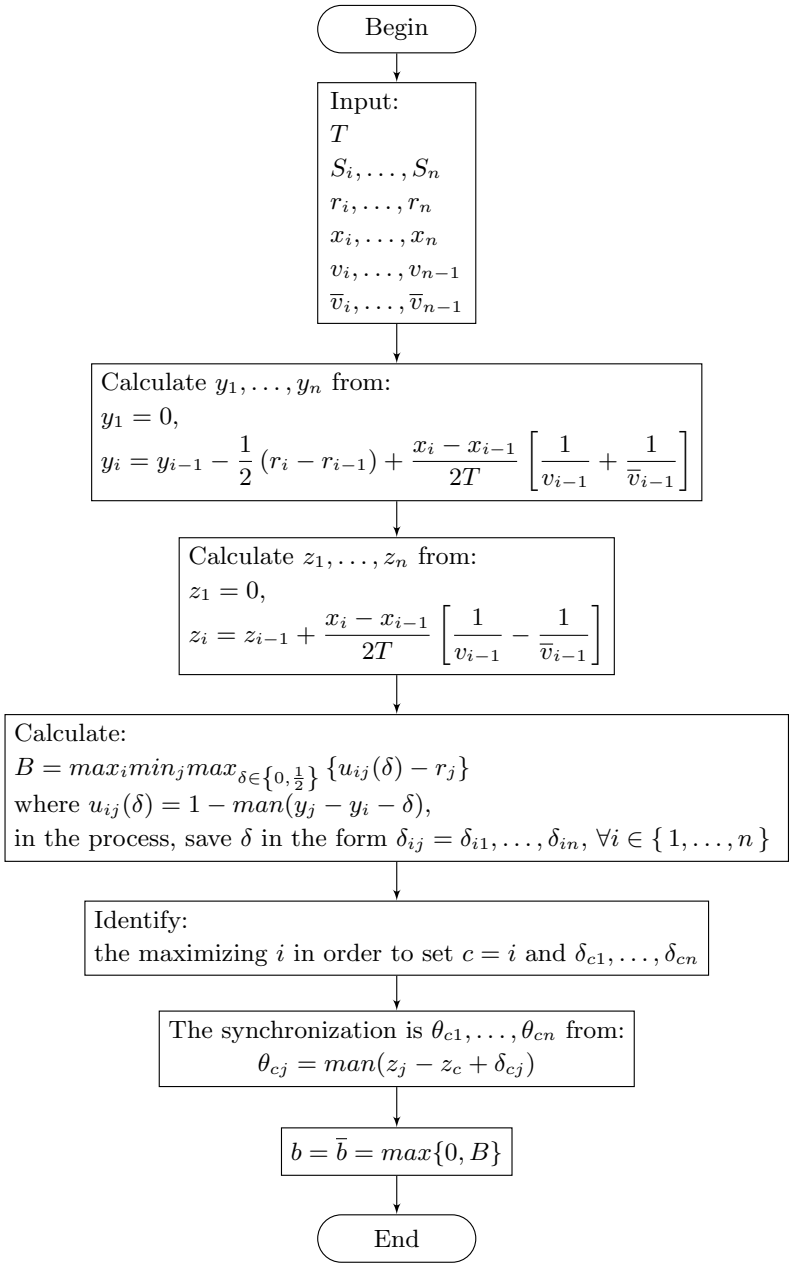
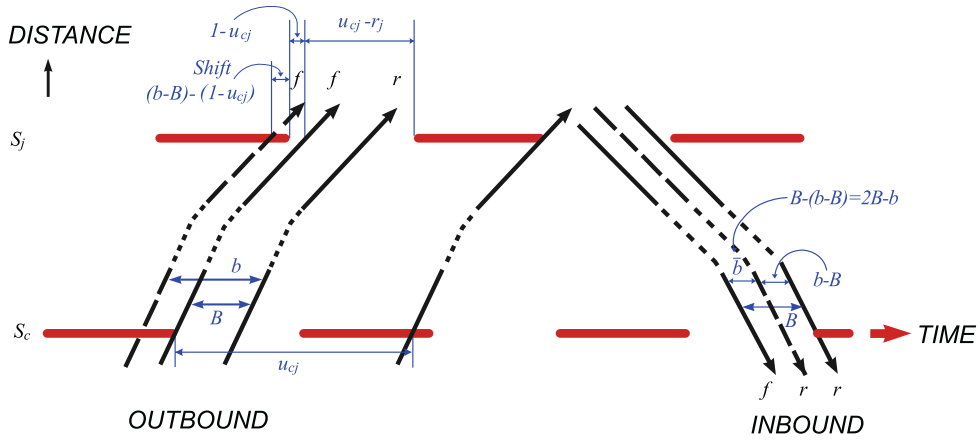


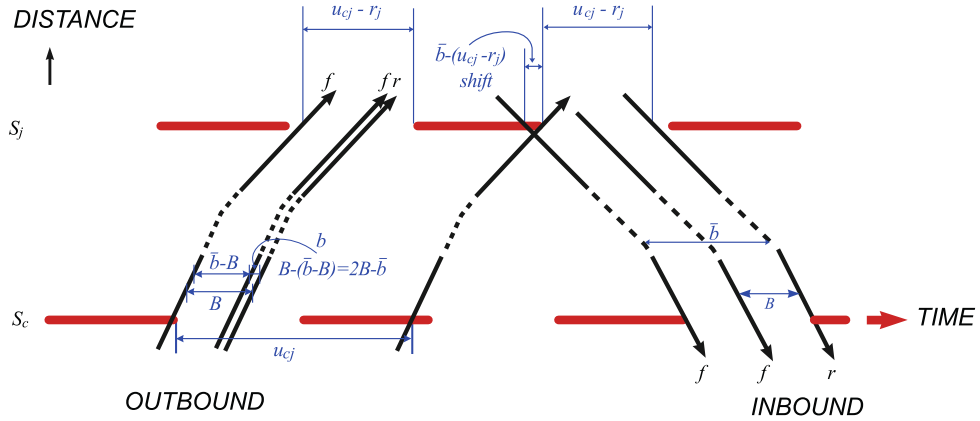
Figure 2.15: SEB Procedure.

- α_j : A phase shift for S_j, in periods.
- $\theta'_{cj} = man[\theta_{cj} - \alpha_j]$: Adjusted phase for S_j, in periods.
- $g = min_i\{1 - r_i\}$: Smallest green time, in periods.

According to Figure 2.16a, suppose we want to move S_c to the left because we want to increase outbound bandwidth from B to b. $\overline{b}$ will decrease as much as b is increased. Furthermore, due the signals limit the movement, the shift can be up to g. Thus, $max\{0, B\} \leq b \leq g$ and $\overline{b} = max\{2B - b, 0\}$. A similar argument applies to increase $\overline{b}$, then $max\{0, B\} \leq \overline{b} \leq g$ and $b = max\{2B - \overline{b}, 0\}$, see Figure 2.16b.



(a)



(b)

Figure 2.16: Geometry for SHIFT Procedure.

If we place on the right-hand side of the S_j red time in Figure 2.16a, it can be seen that the shift for S_j is to left by $\alpha_j = \{(b-B)-(1-u_{cj})\} = max\{u_{cj}-1+b-B, 0\}$. Also, it is true, from Figure 2.16b, that the distance from the front of the inbound green band to the next S_j red on the left is the same than the distance from the rear of the outbound green band to the next S_j red on the right. This is due to Theorem 4 in order to keep the constant $b+\overline{b}$. From Figure 2.16b, it can be seen that to increase $\overline{b}$ is necessary shift S_j by $\alpha_j = max\{\overline{b}+r_j-u_{cj}, 0\}$. □

The theorem produces the following SUB Procedure (Synchronization for Maximal Unequal Bandwidths). Let $P(\overline{P})$ be the platoon length in the outbound (inbound) direction (periods). If $P = \overline{P}$, then the synchronization given by SEB Procedure is accepted. Otherwise, if $P+\overline{P} \leq 2B$ then it is possible to do a shift in order for both platoons to pass through the street without stopping and the bandwidth is changed proportionally to the platoon length if possible, i.e., if $P > \overline{P}$, then $b = min\left\{\dfrac{2BP}{P+\overline{P}}, g\right\}$ and $\overline{b} = [2B-b, 0]$. On the other hand, if $P+\overline{P} > 2B$, the larger platoon is accommodated if possible, the remainder is given to the smaller, of course if there is any remainder, i.e., if $P > \overline{P}$, then $b = min\{P, g\}$ and $\overline{b} = max\{2B-b, 0\}$. We have to consider that if $P \geq 2B$ then $\overline{b}$ will be zero and b will be g, i.e., due to the lack of time to accommodate the platoon in that direction only the outbound traffic will flow. Similar arguments apply if $\overline{P} > P$. The SUB Procedure is summarized in Figure 2.17.

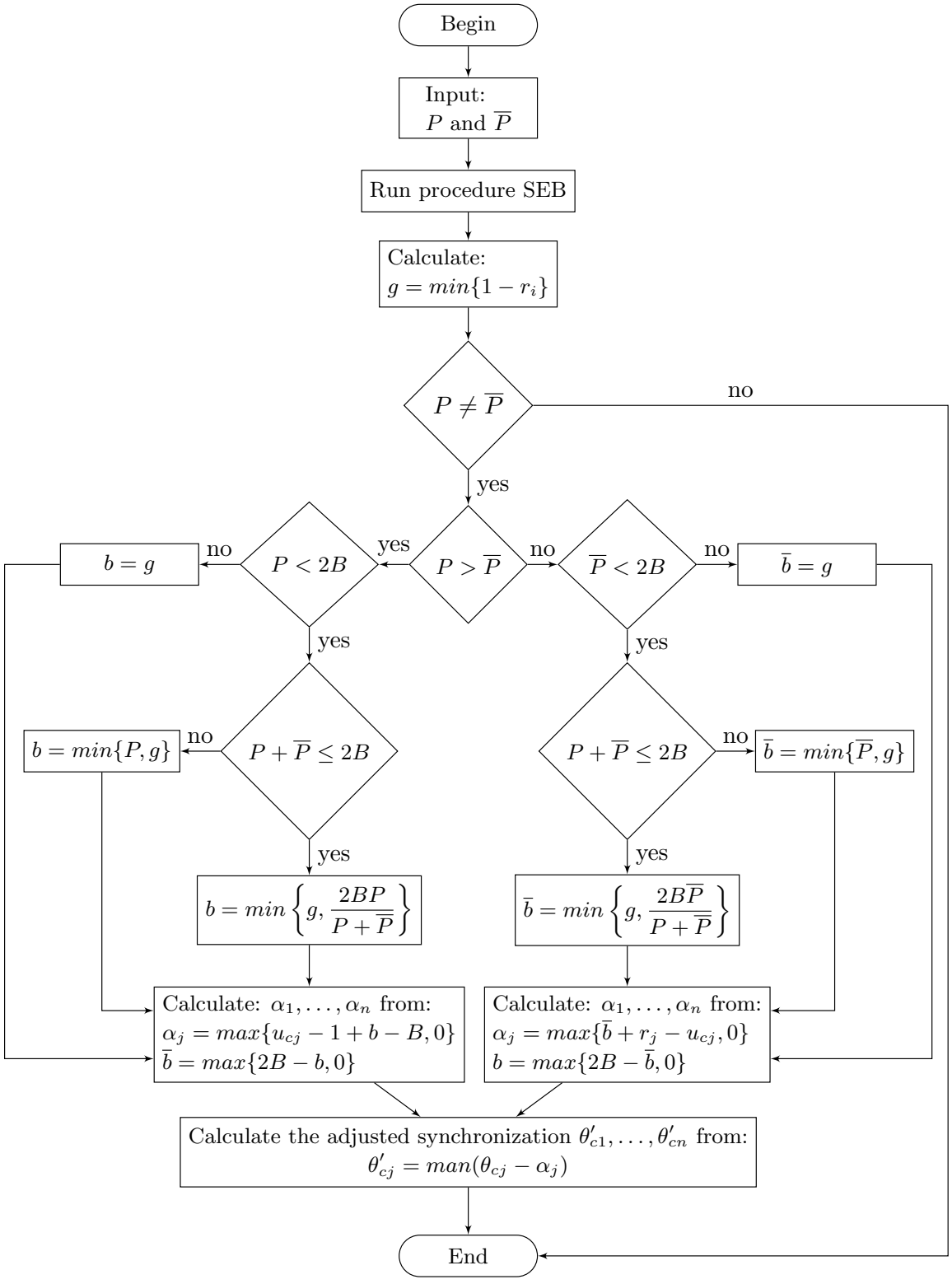


Figure 2.17: SUB Procedure, Synchronization for Maximal Unequal Bandwidths.

2.5.3 Examples

Due to its simple programming language and its wide graphic capacity, MATLAB has been used to code the SEB and SUB procedures and the instance shown in Morgan and Little (1964) (synchronization of the signals on a stretch of Euclid Avenue in Cleveland under off-rush hour conditions) has been a base to produce three different examples. In those cases, and following the original example, the distances are given in feet and velocities in feet/second.

Table 2.4: Input data for Euclid Avenue instance with $P = \overline{P}$.

Signal	1	2	3	4	5	6	7	8	9	10
Distance	0	550	1250	2350	3050	3850	4500	4900	5600	6050
Red time	0.47	0.40	0.40	0.47	0.48	0.42	0.40	0.40	0.40	0.42
OutB velocity	50	50	50	50	50	50	50	50	50	-
InB velocity	50	50	50	50	50	50	50	50	50	-

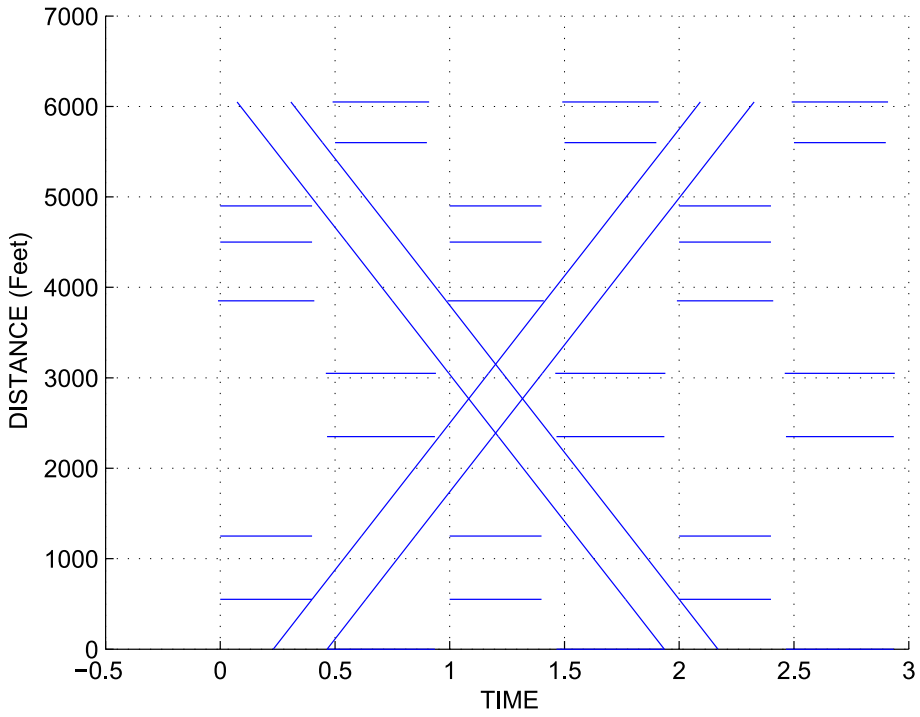


Figure 2.18: Space-time diagram for for Euclid Avenue instance with $P = \overline{P}$.

Table 2.5: Output for Euclid Avenue instance with $P = \overline{P}$.

Signal	1	2	3	4	5	6	7	8	9	10
θ	0.50	0.00	0.00	0.50	0.50	0.00	0.00	0.00	0.50	0.50

OutB bandwidth length	0.23
InB bandwidth length	0.23

Table 2.6: Input data for Euclid Avenue instance with $P \neq \overline{P}$.

Signal	1	2	3	4	5	6	7	8	9	10
Distance	0	550	1250	2350	3050	3850	4500	4900	5600	6050
Red time	0.47	0.4	0.4	0.47	0.48	0.42	0.4	0.4	0.4	0.42
OutB velocity	50	50	50	50	50	50	50	50	50	-
InB velocity	50	50	50	50	50	50	50	50	50	-

OutB platton length	0.30
InB platton length	0.10

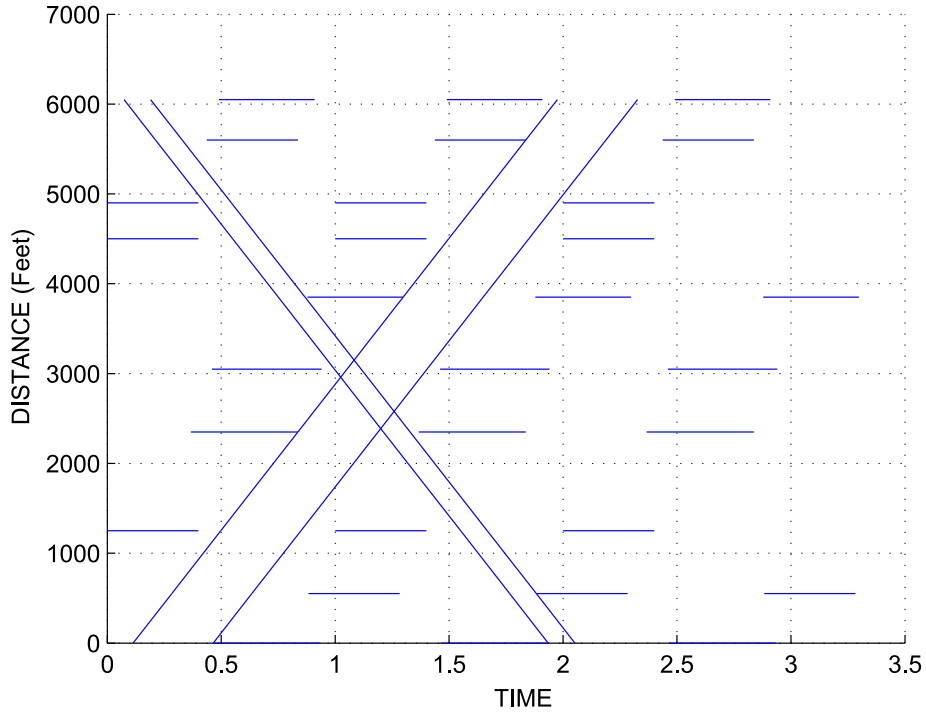


Figure 2.19: Space-time diagram for Euclid Avenue instance with $P \neq \overline{P}$.

Table 2.7: Output for Euclid Avenue instance with $P \neq \overline{P}$.

Signal	1	2	3	4	5	6	7	8	9	10
θ'	0.50	0.88	0.00	0.40	0.50	0.89	0.00	0.00	0.44	0.50
α	0.00	0.12	0.00	0.10	0.00	0.11	0.00	0.00	0.06	0.00

OutB bandwidth length	0.35
InB bandwidth length	0.12

Table 2.8: Input data for Euclid Avenue instance with $P \neq \overline{P}$ and different velocities.

Signal	1	2	3	4	5	6	7	8	9	10
Distance	0	550	1250	2350	3050	3850	4500	4900	5600	6050
Red time	0.47	0.40	0.40	0.47	0.48	0.42	0.40	0.40	0.40	0.42
OutB velocity	50	20	50	50	100	50	10	50	30	-
InB velocity	50	20	5	50	28	50	120	50	20	-

OutB platton length	0.30
InB platton length	0.10

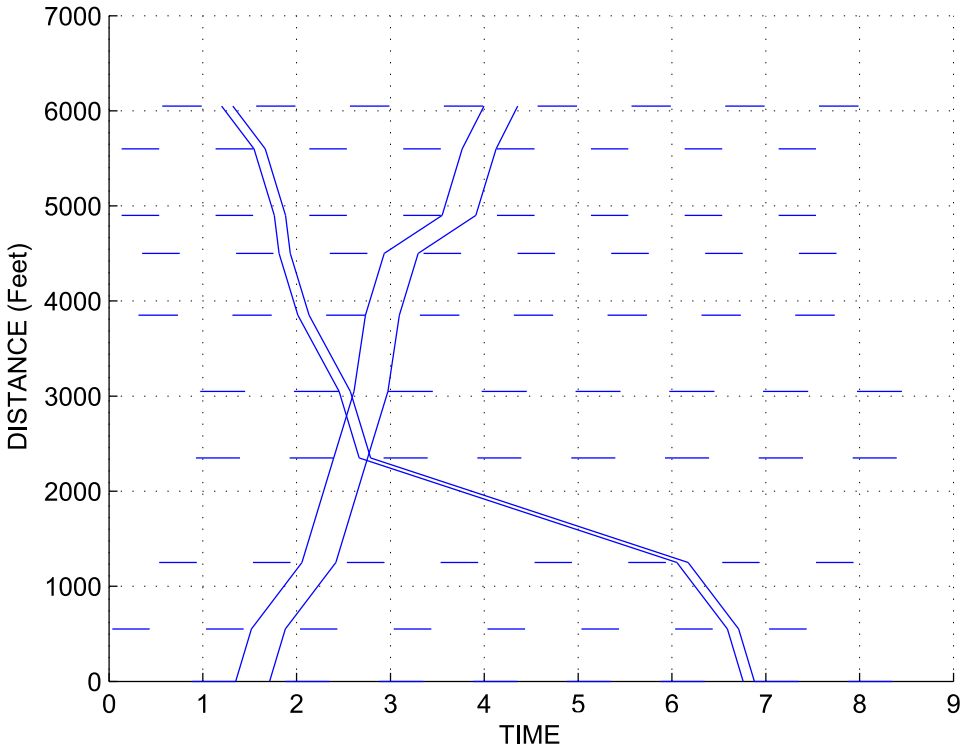


Figure 2.20: Space-time diagram for Euclid Avenue instance with $P \neq \overline{P}$ and different velocities.

Table 2.9: Output for Euclid Avenue instance with $P \neq \overline{P}$ and different velocities.

Signal	1	2	3	4	5	6	7	8	9	10
θ'	0.88	0.00	0.50	0.93	0.98	0.29	0.32	0.10	0.01	0.54
α	0.12	0.00	0.00	0.05	0.00	0.03	0.00	0.00	0.00	0.00

OutB bandwidth length	0.36
InB bandwidth length	0.12

2.5.4 Conclusions

The presented procedures solve efficiently a particular case of the problem of synchronization of traffic lights, the arterial case with common period. It is clear that the same problem can be solved via linear programming by modifying the MAXBAND formulation. However, in addition to its historical importance, the study of this algorithm has led us to a better understanding of the MAXBAND geometry. Afer the presented method was proposed some linear formulations were proposed by Little (1966), but, as the author pointed out, the computational limitations were an issue in the mid-sixties. The extension to the case over a complete network that includes loops was logical.

2.6 An MILP-Based Heuristic with Tabu Search for MAXBAND

In this section we propose a metaheuristic algorithm to solve the MAXBAND model with three variants. Then, we perform computational experiments and show the results.

On a network, the MAXBAND model requires a large number of initial data as well as several variables defined on each segment of an artery, signals and cycles in the cycle basis. Even though the instances presented in Table 2.10 may seem not very large, the formulation has many equalities which contain integer and binary variables. This makes the problem difficult to solve.

Table 2.10: Sizes of some MAXBAND problems.

	Grid Graph $G_{r\times c}$								
	3x3	5x5	6x6	7x7	8x8	9x9	10x10	15x15	20x20
Equalities	16	56	85	120	161	208	261	616	1121
m's	12	40	60	84	112	144	180	420	760
C's	4	16	25	36	49	64	81	196	361
δ's	36	100	144	196	256	324	400	900	1600
Integer variables	52	156	229	316	417	532	661	1516	2721

As will be seen later, a commercial MILP solver is not able to find the optimum for instances greater than 6×6 grid graph within a reasonable time (less than 3 hours). Therefore the use of heuristic methods is a logical alternative to the problem we are studying.

One of the methods of solution for timing traffic lights is that used by TRANSYT, a commercial software that bases its solution on hill-climbing, evolutionary algorithm methods and simulation for modelling the behaviour and interactions of traffic flow. In Ratrout and Reza (2014) and Lu et al. (2014) TRANSYT was compared with other similar software and showed to be more efficient than its competitors in terms of performance index (for example, number of vehicle stops) and determination of the common period length (red plus green light) for each signal on the network. In fact, some authors compare the results of their proposed approaches with those obtained using TRANSYT, see Wünsch (2008) as an example. Regrettably, TRANSYT methodology has not been developed to be adapted to a problem of bandwidth maximization and therefore it is not an option to solve the MAXBAND model. This approach optimizes, instead of the bandwidth, a very complete objective function that involves delay, stops, fuel consumption, etc. thus most of the MAXBAND constraints are explicitly defined in it.

A heuristic alternative for MAXBAND was proposed by Gartner and Stamatiadis (2002). Their approach exploits the intrinsic geometry on the network. One iteration of this method consists of two general stages. The first one solves a subproblem from the original one with

standard MILP techniques. The subproblem is a tree which is chosen because it contains no cycles. Moreover, this tree is not selected randomly as it considers either the streets with higher volume of traffic flow or some other measure of interest. This is why this tree is called *priority arterial subnetwork*. After solving this reduced problem, the integer variables are fixed to the values obtained and used in a second stage to solve the whole problem. The procedure can be repeated if further improvements are required. If the original problem has a large size (dense when using all MAXBAND constraints), then the priority tree is difficult to solve, even though no integer cycle variables are used, the remaining variables could still be many. In fact, in Gartner and Stamatiadis (2002) only two real but small cases were solved, the largest one is an incomplete 4×4 grid graph.

The method we propose is an adaptation of heuristic methods applicable for MAXBAND to try to solve larger instances.

The use of simulation helps the construction of multiple independent instances of a problem to obtain multiple initial feasible solutions, as required in evolutionary algorithms and how it is used in TRANSYT. Instead, the procedure we propose is based on a single feasible solution, as do heuristics such as Simulated Annealing (Kirkpatrick et al., 1983) and Tabu Search (Glover, 1986). We have decided to use a Tabu Search approach because its structure exploits intelligent strategies based on learning procedures that allows a guided search towards the optimal solution. Tabu Search uses a memory structure to generate new solutions from an initial one. These will compete among each other in each iteration of the procedure. Due to memory structure, changes applied in a solution to generate another one are considered in future iterations which, in favour of the diversity of the solutions, we would not like to repeat often. This avoids frequently repeating very complicated problems to be solved within the procedure. We use the variable fixing ideas to obtain new solutions in a short time. The method also involves a local search procedure to intensify the search of good solutions. It is done in three different ways that will be mentioned later.

The MILP-Based Heuristic

As our algorithm is based on tabu search therefore it requires an initial solution whose neighbourhood needs to be explored. Despite many attempts to find a systematic procedure to generate the initial solution, this was not possible (due to the high number of equalities involved). Instead, we take as starting solution the first feasible solution found by the optimization solver (Xpress).

Next, we consider the set of the variables m's $(2rc-r-c)$, δ's $(4rc)$ and C's $(r(c-1)-c+1)$ that must take integer values in the optimal solution. A number of rm, rd and rC variables are chosen randomly from them respectively to be modified later with one of the following procedures. The rest of these integer variables are fixed to the values that they have in the initial solution.

- TSILP-LSF: The values of rm, rd and rC variables are fixed to values within the bounds (2.21), $\{0, 1\}$ and (2.31) respectively.
- TSILP-LSU: The rm, rd and rC variables with values given by a solution become variables again (i.e., unfixed).
- TSILP-LSVNS: TSILP-LSF and TSILP-LSU are applied one after the other.

The current problem is then solved with Xpress to obtain a new feasible neighbour solution. This is repeated in order to generate a set of solutions (*candidate list*) of size *SizeList*. The

list of candidates may contain many infeasible solutions when using TSILP-LSF. If we unfix some variables, the number of infeasible solutions is reduced considerably. Indeed, we have used TSILP-LSU to generate the candidate list. Additionally, in our experience, if a memory structure is applied, procedures such as release (unfix) variables and solve the problem that remains, give a diverse enough set of solutions. Thus, TSILP-LSU is applied on the selected variables rm, rd and rC only if it is not forbidden by a *tabu list*. This list is an array that contains tt values for each variable which represent the number of iterations that they can not be modified. Then, this is sorted in decreasing order by the objective value (i.e., best first) and the the first solution becomes *current solution*. It is saved and the *tabu list* is updated for each integer variable. The last is done by decreasing tt values if they are different from zero and by fixing them to a value $maxtt$ otherwise. Subsequently a greedy local search is applied to the *current solution*. This is simply an iterative application of TSILP-LSVNS (Algorithm 2.6.1), TSILP-LSF (Algorithm 2.6.2) or TSILP-LSU (Algorithm 2.6.3). Finally this can be repeated until a maximum number of iterations is met. See Algorithm 2.6.4.

VNS in TSILP-LSVNS stands for variable neighbourhood search, see Mladenović and Hansen (1997). As can be seen later, the combination of fixing and unfixing random integer variables performs better.

Algorithm 2.6.1 VNS Local Search Procedure (LSVNS).

Let:
S : A problem with best objective function value on a *candidate list*.

Step 1. Choose rm, rd and rC from m's, δ's and C's on S.
Step 2. If $tt = 0$, fix their values with random numbers within their bounds.
Step 3. Solve the instance using an LP solver (Xpress) and set this solution as a *current solution*.
Step 4. Choose other integer variables rm, rd and rC from m's, δ's and C's on the *current solution*.
Step 5. If $tt = 0$, these variables are unfixed.
Step 6. Solve the instance using branch and bound (Xpress).
Step 7. Update the *current solution* only if it is better than the previous one.
Step 8. Repeat the process until the maximum number of iterations is reached.

Algorithm 2.6.2 Fix local search procedure (LSF).

Let:
S : A problem with best objective function value on a *candidate list*.

Steps 1 . . . 3 and 7 . . . 8 are as in LSVNS.

Algorithm 2.6.3 Unfix local search procedure (LSU).

Let:
S : A problem with best objective function value on a *candidate list*.

Steps 4 . . . 6 and 7 . . . 8 are as in LSVNS.

Algorithm 2.6.4 Tabu Search for MAXBAND (TSILP-LS/F/U/VNS).

Let:
P: A complete MAXBAND problem on a grid graph.

Step 1. For all m, δ and C, $tt = 0$ in a *tabu list*.
Step 2. Find the first integer solution for P by Xpress and set it as *current solution*.
Step 3. Create a *candidate list* of size *SizeList*:
1. Choose randomly rm, rd and rC from m's, δ's and C's in *current solution*.
2. If $tt = 0$, unfix these variables.
3. Solve the problem using branch and bound (Xpress).
4. Repeat *SizeList* times.

Step 4. Sort the *candidate list* in decreasing order of the objective function value.
Step 5. Apply procedure LSF, LSU or LSVNS with the best first in the candidate list as input.
Step 6. Set the *current solution* as the best solution in the *candidate list*.
Step 7. Update the *tabu list* for each m, δ and C on the *current solution*:
1. If current $tt = 0$ then $tt = maxtt$, else
2. $tt = tt - 1$.

Step 8. Repeat Steps $3 - 7$ until a maximum number of iterations is reached.

2.6.1 Computational Results

Because we did not have access to real case information we have generated some artificial data as described below. The intervals of random values are based on the small example of five arteries and seven signals provided by Little (1966) which was solved with branch and bound by hand. All random data take the same values for outbound and inbound direction. It must be noted that even small grid graphs are very dense.

Let $U(a, b)$ be a continuous uniform distribution on interval (a, b),

- The lengths of the arcs of the grid follow a distribution $U(140, 600)$ (meters).
- Red times r follow a distribution $U(0.4, 0.6)$ (periods).
- Times to turn left ℓ follow a distribution $U(0.25r, 0.38r)$ (seconds).
- Min/max common period T_{min}/T_{max}, follows a distribution $U(40, 60)/U(90, 110)$ (seconds).
- Limits of velocities lower/upper e/f follow a distribution $U(12, 14)/U(15, 16)$ (meters/second).
- Limits on changes in reciprocal speed lower/upper $1/h/1/g = 0.012/-0.012$ $(\text{meters/second})^{-1}$.
- All τ_{ai}'s and $\overline{\tau}_{ai}$'s were set to 0.
- All weights on the objective function were set to 1.

We used a PC Intel(R) Xeon(R) 3.40GHz 16.0 (RAM). Strictly fundamental cycle basis (Kavitha et al., 2009; Liebchen and Rizzi, 2007) for each graph $G_{r\times c}$ were found with Mathematica version 10.1. The algorithms were coded with Xpress Mosel version 3.4.2 and the solver used was Xpress Optimizer version 24.01.04.

Table 2.11 shows the results for several small grid graph instances generated randomly considering all parameters and variables in model 2.3.1 including the bounds (2.21) and (2.31).

Table 2.11: Computational Results for TSILP procedure (small instances).

		Exact		Global			Global/LS				TSILP-LSF				TSILP-LSU				TSILP-LSVNS			
size	#	OF*	t	iter	sl	tt	iLS	rm	rd	rC	avg	worst	best	avgt	avg	worst	best	avgt	avg	worst	best	avgt
3x3	1	3.22	0	10	5	3	5	2	2	2	2.73	2.71	2.91	6	3.01	3.01	3.02	7	3.00	2.83	3.02	9
				10	5	3	10	2	2	2	2.72	2.71	2.81	8	3.01	3.01	3.02	10	3.02	3.01	3.02	15
				30	10	3	10	4	4	4	2.71	2.71	2.76	33	3.02	3.02	3.02	45	3.02	3.02	3.02	59
				50	10	3	20	4	4	4	2.71	2.71	2.71	73	3.02	3.02	3.02	123	3.02	3.02	3.02	174
	2	3.80	0	10	5	3	5	2	2	2	2.61	1.24	3.76	5	3.48	1.38	3.80	6	3.79	3.67	3.80	8
				10	5	3	10	2	2	2	3.24	1.24	3.77	10	3.78	3.72	3.80	9	3.80	3.76	3.80	13
				30	10	3	10	4	4	4	3.69	3.69	3.69	31	3.69	3.69	3.69	39	3.69	3.69	3.69	52
				50	10	3	20	4	4	4	3.69	3.69	3.69	72	3.69	3.69	3.69	102	3.69	3.69	3.69	145
5x5	3	4.77	7	10	5	3	5	2	2	2	3.57	2.63	4.16	11	4.05	3.43	4.31	14	4.11	3.73	4.29	17
				10	5	3	10	2	2	2	3.86	3.29	4.06	14	4.08	3.52	4.30	17	4.17	3.83	4.35	19
				30	10	3	10	4	4	4	4.16	4.01	4.41	55	4.31	4.18	4.73	96	4.36	4.18	4.73	117
				50	10	3	20	4	4	4	4.14	3.94	4.18	145	4.33	3.95	4.71	257	4.42	4.23	4.72	332
	4	5.20	6	10	5	3	5	2	2	2	3.88	3.75	3.98	7	4.09	3.84	4.43	11	4.24	3.98	4.49	11
				10	5	3	10	2	2	2	3.91	3.74	4.11	14	4.25	3.78	4.77	22	4.21	4.11	4.26	29
				30	10	3	10	4	4	4	4.47	3.97	4.61	58	4.77	4.66	4.86	84	4.74	4.46	4.86	120
				50	10	3	20	4	4	4	4.45	3.97	4.61	145	4.78	4.66	4.86	260	4.83	4.77	4.86	326
6x6	5	5.18	116	10	5	3	5	2	2	2	3.11	2.01	3.52	15	3.51	3.28	3.94	17	3.54	3.18	3.97	20
				10	5	3	10	2	2	2	3.12	2.49	3.48	21	3.49	3.07	3.79	33	3.48	3.07	3.86	39
				30	10	3	10	4	4	4	3.58	3.38	3.73	85	4.07	3.57	4.24	128	4.16	3.91	4.24	152
				50	10	3	20	4	4	4	3.73	3.49	4.18	190	4.23	3.93	4.46	345	4.31	4.12	4.46	420
	6	4.74	1270	10	5	3	5	2	2	2	3.35	1.63	4.16	12	4.18	3.92	4.31	17	4.16	3.92	4.31	19
				10	5	3	10	2	2	2	3.57	1.63	4.15	17	4.19	3.74	4.35	25	4.26	4.10	4.35	36
				30	10	3	10	4	4	4	4.24	4.17	4.33	92	4.32	4.14	4.37	128	4.36	4.35	4.37	152
				50	10	3	20	4	4	4	4.28	4.22	4.33	193	4.36	4.35	4.37	274	4.36	4.35	4.37	295

The meanings of the headers are as follows:

- size: size of the problem.
- #: instance number.
- Exact (Xpress branch-and-bound).
  - OF*: optimal value of the objective function.
  - t: running time (seconds).
- Global (stands for whole procedure).

- iter: number of tabu search iterations.
- sl: size list.
- tt: tenure time in the memory list (iterations).

- Global/LS (stands for whole and local search procedures).
  - iLS: number of iterations of local search.
  - rm, rd, rC: number of variables m's, δ's and C's chosen for the candidate list and the local search.

- TSILP-(LSF, LSU, LSVNS).
  - avg, worst, best: Average, worst and best case objective function value.
  - avgt: average time (seconds).

Each problem has been solved ten times with four different parameters settings. The same number of integer variables rm, rd and rC were used in both of them to create a candidate list and to run the local search procedure. The best objective function values obtained among the different heuristic algorithms are underlined.

For each instances the optimal objective function could be obtained by Xpress in short time. TSILP-LSU and TSILP-LSVNS could meet the optimal for problem #2 and both algorithms performed almost the same in the most cases. See Figure 2.21 as an example, red dots represents the mean OF values.

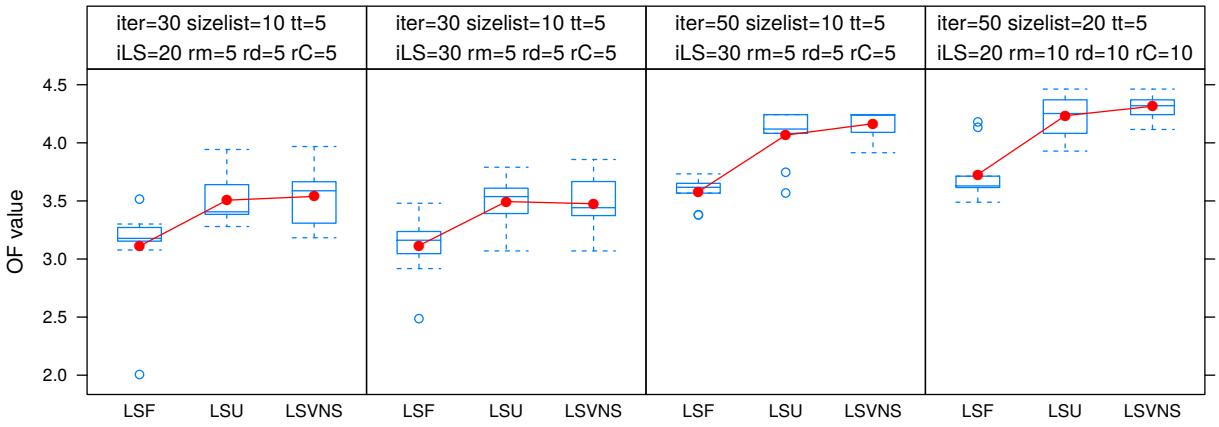


Figure 2.21: Box-Plot of 10 runs for experiments TSILP-LS on $G_{6\times 6}$ (instance 5).

For 5×5, 6×6 sizes problems the increase in the number of iterations, size list and selected random variables produced better results. It is clear that as the problem size increases, the heuristics times are more competitive.

Results for larger instances are shown in Table 2.12. Xpress was not able to find the optimal solution for any of them within a time limit of 3 hours. For these instances it can be observed that TSILP-LSVNS performs better. See Figures 2.22 and 2.23.

The running times include the time needed to find the first initial feasible solution using the standard branch and bound algorithm in Xpress. For instance #13 this time was 9657 seconds, but just 152 for instance #14. The remainder took less than 11 seconds.

In most of the cases the average solution of TSILP-LSVNS improved the average solutions obtained with the first two methods. TSILP-LSF took less time, but almost never found the best objective function value obtained by the other algorithms.

Table 2.12: Computational Results for TSILP procedure (large instances).

		Global			Global/LS				TSILP-LSF				TSILP-LSU				TSILP-LSVNS			
size	#	iter	sl	tt	iLS	rm	rd	rC	avg	worst	best	avgt	avg	worst	best	avgt	avg	worst	best	avgt
7x7	7	30	10	5	20	5	5	5	4.33	4.24	4.34	292	4.51	4.35	4.66	484	4.63	4.56	4.68	546
		30	10	5	30	5	5	5	4.34	4.27	4.46	213	4.51	4.35	4.72	440	4.58	4.36	4.72	520
		50	10	5	30	5	5	5	4.34	4.34	4.35	300	4.56	4.35	4.70	625	4.66	4.53	4.72	815
		50	20	5	30	10	10	10	4.37	4.24	4.55	623	4.67	4.56	4.72	1469	4.70	4.66	4.72	1700
	8	30	10	5	20	5	5	5	3.11	2.88	3.19	175	3.28	3.19	3.46	347	3.40	3.19	3.55	417
		30	10	5	30	5	5	5	3.15	2.88	3.19	162	3.31	3.12	3.48	329	3.41	3.19	3.83	425
		50	10	5	30	5	5	5	3.21	3.19	3.38	399	3.35	3.19	3.55	613	3.63	3.23	3.99	817
		50	20	5	30	10	10	10	3.29	3.19	3.82	581	3.52	3.22	4.30	1113	3.81	3.36	4.39	1323
8x8	9	30	10	5	20	5	5	5	4.08	3.85	4.28	219	4.22	4.20	4.28	398	4.23	4.20	4.28	483
		30	10	5	30	5	5	5	3.99	3.85	4.19	281	4.21	4.20	4.24	715	4.24	4.20	4.28	910
		50	10	5	30	5	5	5	4.04	3.85	4.19	472	4.23	4.20	4.33	806	4.25	4.20	4.33	900
		50	20	5	30	10	10	10	4.22	4.19	4.28	778	4.28	4.20	4.33	1710	4.30	4.24	4.33	1760
	10	30	10	5	20	5	5	5	2.25	2.08	2.84	167	3.07	2.54	3.37	397	3.10	2.75	3.53	524
		30	10	5	30	5	5	5	2.25	2.01	2.84	218	3.03	2.38	3.49	601	3.12	2.77	3.49	727
		50	10	5	30	5	5	5	2.33	2.08	2.96	382	3.35	2.84	3.72	710	3.42	2.84	3.61	1186
		50	20	5	30	10	10	10	3.01	2.68	3.22	567	3.54	3.27	3.92	1345	3.72	3.29	4.10	1688
9x9	11	30	10	5	20	5	5	5	3.33	3.00	3.70	282	4.15	3.61	4.35	524	4.15	3.61	4.31	620
		30	10	5	30	5	5	5	3.40	3.00	4.02	324	4.04	3.61	4.31	620	4.20	3.61	4.49	755
		50	10	5	30	5	5	5	3.43	3.00	3.65	522	4.25	3.61	4.88	893	4.93	4.75	5.01	928
		50	20	5	30	10	10	10	4.14	3.79	4.42	479	4.36	4.26	4.49	1131	4.80	4.38	5.20	1206
	12	30	10	5	20	5	5	5	4.17	4.07	4.31	270	4.31	4.24	4.71	541	4.36	4.24	4.71	646
		30	10	5	30	5	5	5	4.18	4.07	4.31	316	4.25	4.22	4.31	699	4.28	4.24	4.42	848
		50	10	5	30	5	5	5	4.18	4.07	4.42	539	4.26	4.24	4.31	901	4.39	4.24	4.74	1223
		50	20	5	30	10	10	10	4.27	4.12	4.42	658	4.43	4.24	4.56	1324	4.45	4.31	4.61	1698
10x10	13	30	10	5	20	5	5	5	1.63	1.55	1.91	9937	1.85	1.76	1.91	10123	1.86	1.76	1.91	10259
		30	10	5	30	5	5	5	1.75	1.55	1.91	10007	1.84	1.57	1.91	10278	1.89	1.72	1.91	10488
		50	10	5	30	5	5	5	1.71	1.55	1.91	10218	1.86	1.66	1.91	10714	1.91	1.90	1.91	11058
		50	20	5	30	10	10	10	1.89	1.72	1.91	10553	1.91	1.91	1.91	11341	1.94	1.73	2.01	11729
	14	30	10	5	20	5	5	5	2.36	2.30	2.46	474	2.86	2.56	2.97	778	2.91	2.75	2.97	918
		30	10	5	30	5	5	5	2.45	2.30	2.90	554	2.85	2.52	3.03	769	2.94	2.79	3.05	958
		50	10	5	30	5	5	5	2.40	2.30	2.55	648	2.93	2.84	3.05	1407	2.95	2.79	3.05	2016
		50	20	5	30	10	10	10	2.98	2.77	3.05	876	3.13	3.04	3.21	2082	3.20	3.05	3.24	3238

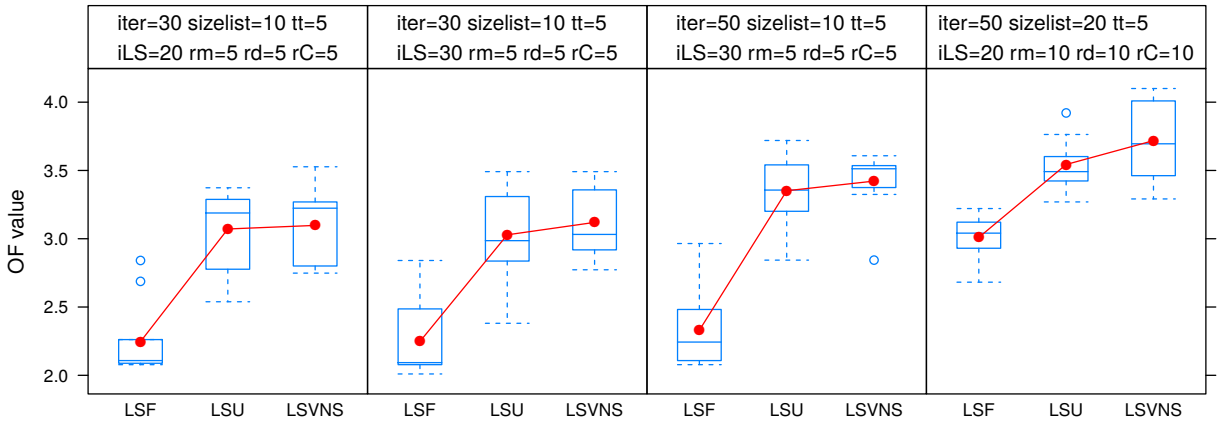


Figure 2.22: Box-Plot of 10 runs for experiments TSILP-LS on $G_{8\times8}$ (instance 10).

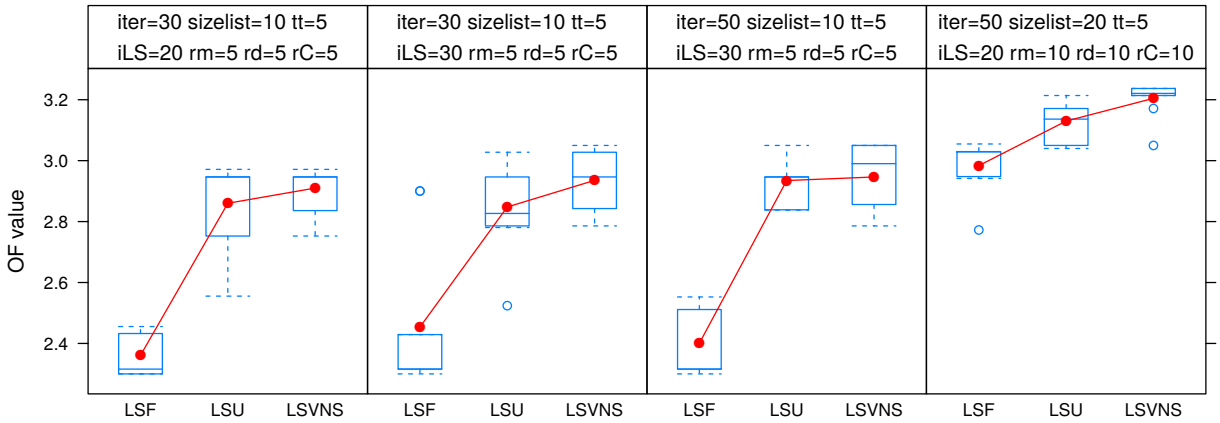


Figure 2.23: Box-Plot of 10 runs for experiments TSILP-LS on $G_{10\times10}$ (instance 14).

Due to the vast amount of initial data, having the integer model ready to use before applying any algorithm takes some time that was not considered in the running times, but for the largest instances tested it averaged 20.73 seconds. This is time used by Xpress to generate the model that will be solved.

The best results were found for the largest instances. In this case the three algorithms, even in the worst cases, reached much better solution than Xpress. Table 2.13 shows the first integer solution (f_OF), the time to meet that solution (t_fs) and the best integer solution after wait a considerable time in relation to those reached in the experiments with the proposed algorithms, 3 hours of running time (t>3h), by Xpress.

Table 2.13: 10×10 instances vs Xpress.

		Exact			TSILP_LSF		TSILP_LSU		TSILP_LSVNS	
size	#	f_OF	t_fs	t>3h	worst	best	worst	best	worst	best
10x10	13	0.14	9657	1.55	1.55	1.91	1.57	1.91	1.72	2.01
	14	0.54	152	2.19	2.30	3.05	2.52	3.21	2.75	3.24

2.6.2 Conclusions

We started with a complete review of MAXBAND. We generalized the integer variables bounds given in Little (1966) to the network case. Cycle integer variables bounds are given as well.

We proposed a heuristic algorithm based on tabu search that takes advantage of the mixed integer linear MAXBAND model to obtain feasible solutions. During the search for a solution, we solve problems in a reduced feasible space, as the proposed algorithm begins with a feasible solution and then we use the obtained integer values to be used in subsequent iterations. The results were better when a serial application of LSF and LSU was done in a local search (VNS). As seen in the experiments, the best results were obtained in the largest instances tested.

In practice, there are many other aspects that require further attention and that we did not take into account in our experiments, such as prioritizing certain arteries with high traffic. This would change the weights in the objective function.

Chapter 3

An MILP model for a Related Problem to TLSP

In this section we introduce The Shortest Path Problem with Traffic Lights (SPPTL). This is a related problem to the Traffic Light Synchronization Problem (TLSP) in the sense that the offsets given by TLSP can be included as constraints on the nodes in a transport network, giving in this way a more realistic problem that restricts the passage of vehicles through them. In this chapter we give a mixed integer linear formulation for this case which takes a flow formulation as a template and makes use of multiple periodic time windows to model the red and green lights through time.

3.1 Introduction

The synchronization of traffic lights can be used as input information to solve other network problems. The Shortest Path Problem (SPP) is one of them, as in this case it is possible to consider movement restrictions due to traffic signals on nodes. If there is a traffic light on every node in a network, they will restrict the movement of vehicles from one node to another in time intervals. In the SPP we are interested in finding the shortest path between a source node and the rest of the nodes, but if there are traffic light constraints on them, they should be in perfect synchronization (as far as possible) to achieve this objective at minimum cost.

To deal with the case mentioned before, we introduce The Shortest Path Problem with Traffic Lights (SPPTL) which is a generalization of the Shortest Path Problem (Dijkstra, 1959) where additional constraints are added on junction streets on a transport network to model the behaviour of traffic lights. Also, on each node there are different directions where a vehicle can go after waiting for a green light.

Chen and Yang (2000) dealt with this subject by particularizing the more general Time-Constrained Shortest Path Problem (TCSPP) which has been studied extensively in the form of vehicle routing problems with time windows (VRPTW), see for instance (Balakrishnan (1993); Russell (1995)). In TCSPP any node has time constraints and the problem is to see when the nodes in the network can be visited with a minimum cost.

The Chen and Yang (2000) method to solve SPPTL the problem is basically a modified Dijkstra's algorithm. It is well known that Dijkstra's algorithm finds a shortest path forest for a simple source node using Fibonacci Heap in its implementation in $O(m+n\ log\ n)$, where n is the number of nodes and m is the number of arcs in a graph, as is pointed out in Fredman and Tarjan (1987). Cheng and Yang found that the complexity of their algorithm for the SPPTL (also by using Fibonacci Heap) is $O(mn\ log\ r + rn^3)$, where r is the number of different time windows

on a node which are necessary to model the periodicity of the red and green times, as will be seen later.

The main aim of this chapter is to provide a network (connection-based) linear formulation for SPPTL. As far as we know, there is no a linear model for the SPPTL in the literature. In the next section we propose a novel MILP formulation which is based in the flow formulation of the $s-t$ shortest path problem on a directed graph. The notation that we follow is almost the same as Chen and Yang (2000) and a case presented in that paper is reproduced with our model using Xpress as an example. Additional examples have been omitted because the proposed model is given only as a base of future studies (e.g., polyhedral structure), such as the case of the flow model for the Shortest Path Problem, which does not compete with a good implementation of the Dijkstra algorithm in large instances.

We start by defining the notation for SPPTL in Section 3.1.1 to then introduce an MILP model in Section 3.2. We tested the model in Section 3.3 by running an example that helps to understand the notation much better. In Section 3.4 we give final conclusions and remarks.

3.1.1 Notation for SPPTL

Let us consider a directed graph $D(V, A)$ with $V = \{1, 2, \ldots, n\}$ a set of vertices (nodes) and A a set of arcs (i, j) where i and j are in V. On each arc (i, j) a function $t_{ij} : A \to \mathbb{R}_+$ is defined and denotes the travel time from node i to node j. There are synchronized traffic lights constraints at each node i which can be seen as a periodic time windows sequence. Indeed, if each time interval corresponding to a change of light at a traffic light on node i is denoted by k, then the set $\Theta_i = \{1, 2, \ldots, |\Theta_i|\}$ represents a sequence of indices of time windows of the form $\tau_i^k = [a_i^k, b_i^k]$, where a_i^k and b_i^k are lower and upper bounds for the time window k at node i. In addition, each time window τ_i^k has associated a set N_i^k with node-triplet elements of the form $\langle h, i, g\rangle$ that means that the kth time window at node i allows one to visit node g through node h.

The problem is to find a shortest path between a source node $s \in V$ and a sink node $t \in V$ that considers the synchronized traffic lights constraints at each node $i \in V$.

3.2 The Linear Model for SPPTL

The formulation should consider multiple time windows due to the sets N_i^k whose elements are repeated as there are changes in the lights of traffic lights periodically. Nevertheless, as far as we know, there are not many applications that consider periodicity in time windows, except those that talk about vehicle routing, see Mesquita et al. (2013). We have taken the base of a typical Shortest Path formulation with Time Windows, and we have extended it to use several of them. See Linear Model 3.2.1.

- Sets:

  V : Set of nodes of G

  A : Set of arcs of G

  Θ_i : Set of indices for time windows of node i

  $PRED_i^k = \{h \mid \langle h, i, g\rangle \in N_i^k\}$

  $SUCC_i^k = \{g \mid \langle h, i, g\rangle \in N_i^k\}$

LM 3.2.1 A Linear Program for SPPTL

$$\text{Minimize} \qquad T(t) \tag{3.1}$$

subject to:

$$\sum_{g:(i,g)\in\delta_i^+} x_{ig} \; - \sum_{h:(i,h)\in\delta_i^-} x_{hi} = \begin{cases} 1; & i = s \\ 0; & i \in V \setminus \{ s,t \} \\ -1; & i = t, \end{cases} \qquad \forall i \in V, \tag{3.2}$$

$$x_{ij}(T_i + t_{ij} - T_j) \leq 0 \;\equiv\; T_i + t_{ij} - T_j \leq M(1 - x_{ij}), \qquad \forall (i,j) \in A, \tag{3.3}$$

$$a_i^k y_i^k \leq T_i \leq b_i^k + M(1 - y_i^k), \qquad \forall i \in V,\ \forall k \in \Theta_i, \tag{3.4}$$

$$y_i^k - \sum_{j\in PRED_i^k} x_{ji} \leq 0, \qquad \forall i \in V \setminus \{ s \},\ \forall k \in \Theta_i, \tag{3.5}$$

$$y_i^k - \sum_{j\in SUCC_i^k} x_{ij} \leq 0, \qquad \forall i \in V \setminus \{ t \},\ \forall k \in \Theta_i, \tag{3.6}$$

$$x_{ij} - \sum_{k\in\Theta_j} y_j^k \leq 0, \qquad \forall (i,j) \in A, \tag{3.7}$$

$$x_{ij} \in \{ 0,1 \}, \qquad \forall (i,j) \in A, \tag{3.8}$$

$$y_i^k \in \{ 0,1 \}, \qquad \forall i \in V,\ \forall k \in \Theta_i, \tag{3.9}$$

$$T(i) \in \mathbb{R}_+, \qquad \forall i \in V. \tag{3.10}$$

- Parameters:

  a_i^k, b_i^k : Lower and upper bounds for the time window k at node i

  t_{ij} : Time associated with arc (i,j)

- Variables:

  $x_{ij} = \begin{cases} 1, & \text{if node } j \text{ is visited after node } i, \\ 0, & \text{else} \end{cases}$

  $y_i^k = \begin{cases} 1, & \text{if the time window } k \in \Theta_i \text{ is selected at the node } i, \\ 0, & \text{else} \end{cases}$

  $T(i)$: Departure time at node i

About the constraints:

- (3.2) : Flow conservation constraints.
- (3.3 – 3.4) : The typical sets of constraints to ensure departure time from node i plus travel time to node j is not greater than departure time from node j. Also, if we are in node i then the departure time from i must be inside its time window interval.
- (3.5 – 3.6) : Both inequalities can be written as the next single form:

$$2y_i^k \leq \sum_{h \in PRED_i^k} x_{hi} + \sum_{g \in SUCC_i^k} x_{ig} \qquad ; \forall i \in V \setminus \{ s, t \}, \ \forall k \in \Theta_i$$

This means that if y_i^k takes value 1 then must be chosen a predecessor and successor of i from $PRED_i^k$ and $SUCC_i^k$ respectively.

- (3.7) : Ensure that if any $x_{ij} = 1$ then at least one time window k must be picked from node j, i.e., one y_j^k must take the value 1.
- In the objective function 3.1 we are just minimizing the departure time at sink node t, because this is the total time consumed by a vehicle passing through the network.

3.3 An Illustrative Example

Let $D(V, A)$ be a directed graph (see Figure 3.1) with:

$V = \{ s, A, B, C, D, t \}$,
$A = \{ (s, A), (s, B), (A, B), (A, C), (A, D), (B, C), (B, D), (C, t), (D, C), (D, t) \}$.

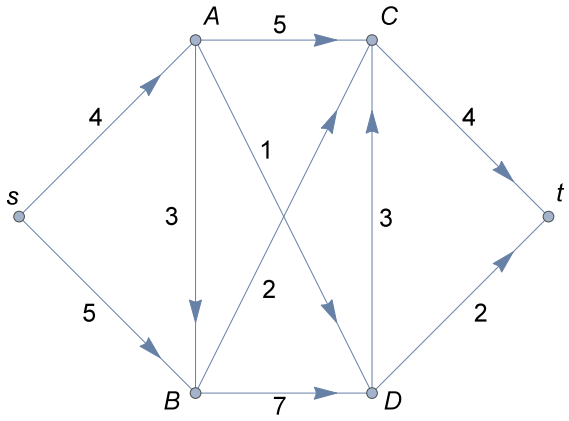


Figure 3.1: The directed graph $D(V, A)$.

On each arc (i, j) of D, t_{ij} is the distance (in time units) to travel from i to j. Furthermore, the possible movements to visit a node g through node a h being at node i ($\langle h, i, g \rangle$) are given by the sets:

$N_A^1 = \{ \langle s, A, D \rangle \}$, $N_A^2 = \{ \langle s, A, B \rangle, \langle s, A, C \rangle \}$,

$N_B^1 = \{ \langle s, B, C \rangle, \langle s, B, D \rangle \}$, $N_B^2 = \{ \langle A, B, C \rangle, \langle s, B, D \rangle \}$,

$N_C^1 = \{ \langle A, C, t \rangle \}$, $N_C^2 = \{ \langle B, C, t \rangle, \langle D, C, t \rangle \}$, $N_C^3 = \{ \langle A, C, t \rangle \}$,

$N_D^1 = \{ \langle A, D, C \rangle, \langle A, D, t \rangle \}$, $N_D^2 = \{ \langle B, D, C \rangle, \langle B, D, t \rangle \}$, $N_D^3 = \{ \langle A, D, C \rangle, \langle A, D, t \rangle \}$,
$N_D^4 = \{ \langle B, D, C \rangle, \langle B, D, t \rangle \}$,

and their respective time windows are:

$\tau_A^1 = [0, 5]$, $\tau_A^2 = [5, 8]$,

$\tau_B^1 = [2, 4]$, $\tau_B^2 = [4, 8]$,

$\tau_C^1 = [3, 5]$, $\tau_C^2 = [5, 9]$, $\tau_C^3 = [9, 11]$,

$\tau_D^1 = [2, 4]$, $\tau_D^2 = [4, 6]$, $\tau_D^3 = [6, 8]$, $\tau_D^4 = [8, 10]$.

Then we have $\Theta_A = \{ 1, 2 \}$, $\Theta_B = \{ 1, 2 \}$, $\Theta_C = \{ 1, 2, 3 \}$, $\Theta_D = \{ 1, 2, 3, 4 \}$.

As an example, if a vehicle arrives to node B from node A at time 7 then it can go only to node C, as the corresponding node-triple for the time window $\tau_B^2 = [4, 8]$ is $\langle A, B, C \rangle$.

The problem is finding a shortest path from s to t. Note that $N_C^1 = N_C^3$, $N_D^1 = N_D^3$ and $N_D^2 = N_D^4$. This is due to the periodicity of the time windows as a vehicle arrives and leaves a node on the optimal path at some time; in this case the example starts from the time zero on s and the time increase while passing through the graph. Then, for C and D we have decided to repeat the node-triplets in order to allow the arrivals to go into a time window, as the time is expected to be big if it is decided to cross them. Indeed, the time windows can be repeated as much as we want to be included in the input data of the lineal model.

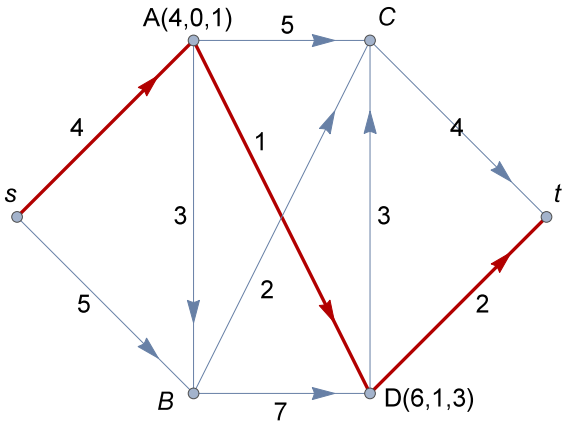


Figure 3.2: $D(V, A)$ with the shortest path from s to t.

The solution by the Linear Model 3.2.1 is given in the Figure 3.2. On the nodes A and D the triplet (a, b, c) represents:

$$(\text{Departure time } T(i), \text{Waiting time on } i, \text{Time windows } \tau_i^k \text{ used}).$$

The final path is $Pd(s, t)$ is $s \to A \to D \to t$, $x_{sA} = x_{AD} = x_{Dt} = 1$ and $y_A^1 = y_D^3 = 1$, the remaining variables are zero.

Consider a modified version of the example previously shown:

- Just as before, let t_{ij} be the associate time on arc(i, j), set $t_{DC} = 1$, $t_{Ct} = 1$ and $t_{Dt} = 10$.
- Change N_D^3 and N_D^4 by $N_D^3 = \{ \langle A, D, t \rangle \}$ and $N_D^4 = \{ \langle A, D, C \rangle, \langle A, D, t \rangle \}$ respectively.

The solution after run our model is:

$Pd(s,t) = s \to A \to D \to C \to t$, $x_{sA} = x_{AD} = x_{DC} = x_{Ct} = 1$ and $y_A^1 = y_D^4 = y_C^2 = 1$, the remaining variables are zero. Also, the triplets with the same meaning as before were $A(4,0,1)$, $D(8,3,4)$ and $C(9,0,2)$.

This example shows that even if a time window in one node allows to go to another one, the model chooses to wait in favour of time minimization ($t_{Dt} > t_{DC} + t_{Ct}$), as on node D the vehicle had to wait for 3 units of time even though it could go directly to sink node t.

3.4 Conclusions, Remarks and Future Work

In this chapter we defined the SPPTL and proposed a novel MILP model based on the classical SPP formulation with periodic time windows. An instance presented in Chen and Yang (2000) was solved. Unlike the modified Dijkstra's algorithm proposed by Cheng and Yang, our model does not give as a result a minimum cost arborescence, but a node-to-node path. The arborescence can be obtained by using the linear model $n-2$ times.

As mentioned in Section 3.1 the method proposed in Chen and Yang (2000) is a modified Dijkstra's algorithm. We would like to highlight that it suggests that other shortest path algorithms can be modified in a similar way, such as Bellman-Ford or Floyd-Warshall. For example, Bellman-Ford can be changed such that it takes into account the multiple time windows that the problem requires with the same limitations (running time) and the same advantages (use of negative arcs) that we can find in the version without traffic lights.

Finally, we would like to point out that a modelling alternative for SPPTL, which has not been studied in this work, is to use a time-space network scheme. In this approach a time-space plane is considered which unlike the connection-based approach, possible movements between two nodes in SPPTL are modeled by directed arcs that are located through time to connect intermediate nodes whenever the light of a traffic signal allows the movement. Models based on this scheme have been successfully studied in scheduling problems, we refer the works of Hane et al. (1995) and Kliewer et al. (2006) for fleet assignment and multi-depot bus scheduling respectively. An application for vehicle routing problem with time windows can be found in Mahmoudi and Zhou (2016). In the case of SPPTL, the representation of the behaviour of traffic lights as periodic time windows suggests that this approach could be suitable to minimize the number of restrictions or parameters that the proposed model contains, for example the Big-M used in the constraints 3.3 and 3.4 could be discarded, since the time windows are implicitly represented in the time-space plane. Our future work on this topic goes this way, we are looking for the minimization of the proposed formulation by studying different ways of representing the problem.

Part II

Simple Plant Location Problem with Order

Chapter 4

A Lagrangean Relaxation Algorithm for the Simple Plant Location Problem with Order

The focus of this chapter is to develop a Lagrangean relaxation algorithm to solve the Simple Plant Location Problem with Order (SPLPO). In particular, this chapter covers the theory about Lagrangean and semi-Lagrangean relaxation and finds similarities in their use with the case without order. As can be seen in the literature, these methods have been successful when have been applied to Simple Plant Location Problem (SPLP), so we have exploited their properties to be implemented to SPLPO. The results obtained by using subgradient and dual-ascent methods to solve the dual programs for both relaxation are put together in one simple procedure that additionally uses a heuristic procedure to find feasible solution as fast as possible. Results of numerical experiments are carried out to show how good the procedure is.

4.1 Introduction

The Simple Plant Location Problem with Order (SPLPO) is a variant of the Simple Plant Location Problem (SPLP), where the customers have preferences over the facilities which will serve them. In particular, customers define their preferences by ranking each of the potential facilities.

Let $I = \{1, \dots m\}$ be a set of customers and $J = \{1, \dots n\}$ a set of possible sites for opening facilities. Unit costs $c_{ij} \geq 0$ for supplying the demand of customer i from facility j and costs $f_j \geq 0$ for opening a facility at j are also considered. It is said that k is *i-worse* than j if customer i prefers facility j to k and it is written as $k <_i j$. We define $W_{ij} = \{k \in J \mid k <_i j\}$ as the set of facilities k strictly *i-worse* than j, its complement as $\overline{W_{ij}}$ and $W_{ij} \cup \{j\}$ as W'_{ij}. Let x_{ij} be a decision variable that represents the fraction of the demand required by customer i and supplied by facility j. Let y_j be a binary variable such that $y_j = 1$ if a facility is open at location j and $y_j = 0$ otherwise.

The SPLPO formulation (Cánovas et al., 2006) is as follows:

LM 4.1.1 Simple Plant Location Problem with Order

$$\text{Min} \quad \sum_{i\in I}\sum_{j\in J} c_{ij}x_{ij} + \sum_{j\in J} f_j y_j,$$

$$\text{subject to} \quad \sum_{j\in J} x_{ij} = 1, \qquad \forall i \in I, \tag{4.1}$$

$$x_{ij} \leq y_j, \qquad \forall i \in I, \forall j \in J, \tag{4.2}$$

$$\sum_{k\in \overline{W_{ij}}} x_{ik} \geq y_j, \qquad \forall i \in I, \forall j \in J, \tag{4.3}$$

$$x_{ij} \geq 0, \qquad \forall i \in I, \forall j \in J, \tag{4.4}$$

$$y_j \in \{0,1\}, \qquad \forall j \in J. \tag{4.5}$$

Equalities (4.1) ensure that every customer i will be supplied by exactly one facility j, they are called *assignment constraints*. Constraints (4.2) ensure that if a customer i is supplied by a facility j then j must be opened, they are usually called *variable upper bounds* (VUBs). Inequalities (4.3) model the customers' preference orderings and they were first presented in Hanjoul and Peeters (1987).

Since no capacities are considered and the model minimizes the number of open facilities y's, the demand of a customer i can always be covered completely by one single facility. Therefore, we can guarantee that there is an optimal solution where values variables x_{ij} are in $\{0,1\}$.

There are in the literature many studies on how to solve the SPLP using different methods and Lagrangean relaxation is one of the most successful. For example, in Cornuéjols et al. (1977), the authors presented an application of Lagrangean relaxation to solve the SPLP in the context of location of bank accounts. They also proposed a heuristic algorithm and studied the Lagrangean dual to obtain lower and upper bounds. They also provided a bound for the relative error of these methods. Beltrán et al. (2006) suggested, defined and applied the technique of semi-Lagrangean relaxation to the p-median problem. Some years later the study was extended to the SPLP, obtaining very good results. The method basically takes advantage of the linear formulation of the problems. First, it splits equality constraints $Ax = b$ into $Ax \leq b$ and $Ax \geq b$, and then relaxes the second one. The new model has the same objective value as the original problem (it closes the duality gap), but with the cost of making the new problem more difficult to solve. However, it has some properties that can be exploited. A summary of this method will be reviewed later. Another reduced form of this method was proposed by Monabbati (2014), which called it a surrogate semi-Lagrangean relaxation. A new algorithm for the dual problem using Lagrangean heuristics for both the original and surrogated version of the semi-Lagrangean relaxation can be found in Jörnsten (2016). Additional information about Lagrangean techniques and heuristics applied to location problems can be found in Guignard and Opaswongkarn (1990) and Wollenweber (2008) with works in the context of limited supply capacity of the facilities and supply chain management respectively.

The SPLPO has been studied much less and the main results are on finding new valid inequalities to strengthen the original formulation. Cánovas et al. (2006) provided a new family of valid constraints which were used in combination with a preprocessing analysis. Another, more general, family of valid inequalities can be found in Vasilyev et al. (2013) with a polyhedral study. A branch and cut method was proposed by Vasilyev and Klimentova (2010).

Since these papers use exact methods that struggle to solve large instances, the aim of this work is to develop a procedure that allows to solve the SPLPO efficiently in a heuristic way by using Lagrangean and semi-Lagrangean relaxation techniques. We propose a variable fixing heuristic that uses a Lagrangean relaxation output as the starting point of a semi-Lagrangean relaxation to find good feasible solutions (often the optimal solution).

We start with Section 4.2.1 where we give a review of Lagrangean and semi-Lagrangean relaxation. The SPLPO version of those models is shown in Section 4.3, 4.3.1, 4.4 and 4.4.1, along with the methods that will be used later to solve their respective dual problems. The complete procedure that we propose is presented in Section 4.5, where all the algorithms given in previous sections will be gathered to build a heuristic procedure that uses just few parameters to be set up. In Section 4.6 the whole method will be tested over a group of large instances. Some conclusions are given in Section 4.7.

4.2 Preliminaries

In this section we review the main results of Lagrangean and semi-Lagrangean relaxation.

4.2.1 A Review of Lagrangean Relaxation

Let P be an original problem of the form:

$$min_x\{f(x) = cx \mid Ax \leq b, Cx \leq d, x \in X\}$$

with:

- The set X could contain integrality constraints.
- The family of constraints $Ax \leq b$ will be assumed to be complicated. (i.e., the problem (P) without them is easier to solve).

The family $Ax \leq b$ can be placed in the objective function with coefficients vector (Lagrangean multipliers) λ which work as penalties when they are violated.

Let $v(p)$ be the optimal objective function value for a particular problem p.

Definition 6 (Lagrangean relaxation. Geoffrion (1974))**.** *The Lagrangean relaxation problem $LR(\lambda)$ related to (P) with multipliers λ is:*

$$LR(\lambda) = min_x\{f(x) + \lambda(Ax - b) \mid Cx \leq d, x \in X, \lambda \geq 0\} \tag{4.6}$$

Theorem 7 (Geoffrion (1974))**.** *For all $x \in X$ feasible for P and any $\lambda \geq 0$, $f(x) + \lambda(Ax - b) \leq f(x)$ (lower bound). Therefore, $v(LR(\lambda)) \leq v(P)$.*

Proof. Since that $x \in X$ is feasible for P then $Ax - b \leq 0$. □

The best multiplier λ is the one that produces the best lower bound for P.

Definition 7 (Lagrangean dual problem LD)**.**

$$LD_\lambda = max_{\lambda \geq 0} min_x\{f(x) + \lambda(Ax - b) \mid Cx \leq d, x \in X, \lambda \geq 0\} = max_{\lambda \geq 0} LR(\lambda)$$

Theorem 8 (Geoffrion (1974))**.** *The objective function in $LR(\lambda)$ is a piecewise linear concave function of λ over its domain.*

Proof. For convenience suppose that P is feasible and the set of feasible solutions of $LR(\lambda)$ is finite. In this case, since $LR(\lambda) = -\lambda b + min_x\{(c + \lambda A)x \mid Cx \leq d, x \in X, \lambda \geq 0\}$, it can be seen that $LR(\lambda)$ is the minimum of a finite number of affine functions. Therefore, it is a concave function of vector λ. □

The most used methods to solve LD_λ use the definition of subgradient due to piecewise nature of the function in $LR(\lambda)$.

Definition 8. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a concave function. The vector s is a subgradient of f at point $\tilde{x}$ if $f(x) - f(\tilde{x}) \leq (x - \tilde{x})s$, for all $x \in \mathbb{R}^n$.*

Theorem 9. *Let x^* be an optimal solution for $LR(\lambda^*)$, then $Ax^* - b$ is a subgradient of $LR(\lambda)$ at point λ^*.*

Proof.

$$\begin{aligned} v(LR(\lambda)) &\leq cx^* + \lambda(Ax^* - b) \\ &= cx^* + \lambda^*(Ax^* - b) + (\lambda - \lambda^*)(Ax^* - b) \\ &= v(LR(\lambda^*)) + (\lambda - \lambda^*)(Ax^* - b), \end{aligned}$$

and it is true for any $\lambda \geq 0$. □

The next results are three important facts about Lagrangean relaxation. More details can be found in Geoffrion (1974), also see Conforti et al. (2014), Fisher (2004) and Guignard (2003).

Theorem 10 (Geoffrion (1974))**.** *If $\{x \mid Ax \leq b, x \geq 0, x \in conv\{x \mid Cx \leq d, x \in X\}\}$ is not empty, then*

$$LD_\lambda = min\{cx \mid Ax \leq b, x \in conv\{x \mid Cx \leq d, x \in X\}\}$$

Proof. Let $Fx \leq g$ be such that $\{x \mid Fx \leq g\} = conv\{x|Cx \leq d, x \in X\}$. Then

$$\begin{aligned} LR(\lambda) &= min_x\{cx + \lambda(Ax - b)|Fx \leq g, x \in X, \lambda \geq 0\} \\ &= max\{\lambda b + \mu g \mid \mu F \geq -c - \lambda A, \lambda \geq 0, \mu \geq 0\} \quad \text{(by duality)} \end{aligned}$$

Therefore, its associated Lagrangean dual problem is

$$\begin{aligned} LD'_\lambda &= max_\lambda LR(\lambda) \\ &= max\{\lambda b + \mu g \mid \lambda A + \mu F \geq -c, \lambda \geq 0, \mu \geq 0\}, \end{aligned}$$

which is the dual of $LD_\lambda = min\{cx \mid Ax \leq b, Fx \leq g\}$ with the same optimal value because the primal is feasible.

□

Corollary 1 (Geoffrion (1974))**.** *Let $LP(P)$ be the problem P without integrality constraints (linear relaxation).*

$$v(LP(P)) \leq v(LD_\lambda) \leq v(P)$$

Corollary 2 (Geoffrion (1974))**.** *$v(LP(P)) = v(LD_\lambda)$ whenever $v(LP(LD_\lambda)) = v(LD_\lambda)$ (integrality property). For pure integer programming, this is true if the matrix of coefficients of $Cx \leq d$ in $LR(\lambda)$ is unimodular and d is an integral vector.*

4.2.2 A Review of Semi-Lagrangean Relaxation

Consider the problem P':

$$min_x\{f(x) = cx \mid Ax = b, x \in X\}$$

with:

- The set X contains integrality constraints.
- A, b and c are non-negative.

Definition 9 (Semi-Lagrangean relaxation. Beltrán et al. (2006))**.** *The semi-lagrangean relaxation problem $SLR(\lambda)$ related to P' with multipliers λ is:*

$$SLR(\lambda) = min_x\{f(x) + \lambda(b - Ax) \mid Ax \leq b, x \in X, \lambda \geq 0\} \tag{4.7}$$

In (4.7), $Ax \geq b$ has been relaxed with a vector multiplier λ and $Ax \leq b$ has been kept as a constraint after splitting $Ax = b$.

Definition 10 (Semi-Lagrangean dual problem SLD_λ, Beltrán et al. (2006))**.**

$$SLD_\lambda = max_{\lambda \geq 0} SLR(\lambda)$$

Theorem 11 (Beltrán et al. (2006))**.**

$$v(LP(P')) \leq v(LD_\lambda) \leq v(SLD_\lambda) \leq v(P')$$

Proof. If for the same problem, the same equality constraint is relaxed and semi-relaxed, then $v(LR(\lambda)) \leq v(SLR(\lambda))$, because $SLR(\lambda)$ is more constrained. The other relations are the same as given in Corollary 1. □

The theorem can be extended to the case of a problem where inequalities and equalities constraints are relaxed and semi-relaxed respectively, i.e. these relations are constraints dependant.

Theorem 12 (Beltrán et al. (2006))**.** $v(P') = v(SLD_\lambda)$

Proof. Applying the Theorem (10) to problem P' yields:

$$\begin{aligned} SLD_\lambda &= min_x \{cx \mid Ax \geq b, x \in conv\{x \mid Ax \leq b, x \in X\}\} \\ &= min_x \{cx \mid Ax \geq b, Ax \leq b, x \in X\} \quad \text{(by theorems (4.1 and 4.3) in Conforti et al. (2014))} \\ &= min_x \{cx \mid Ax = b, x \in X\}. \end{aligned}$$

□

Theorem 13 (Beltrán et al. (2006))**.** *The objective function in $SLR(\lambda)$ is concave, non-decreasing on its domain and $b - Ax$ is a subgradient at point λ.*

Proof. The function is concave by Theorem 8. Let λ and λ' be such that $\lambda \geq \lambda'$ and let x^*_λ and $x^*_{\lambda'}$ be two optimal solutions of SLR at λ and λ' respectively. Then:

$$
\begin{aligned}
v(SLR(\lambda)) &= cx^*_\lambda + \lambda(b - Ax^*_\lambda) \\
&= cx^*_\lambda + \lambda'(b - Ax^*_\lambda) + (\lambda - \lambda')(b - Ax^*_\lambda) \quad (\text{since } \lambda - \lambda' \geq 0 \text{ and } Ax^*_\lambda \leq b \text{ must be held}) \\
&\geq cx^*_\lambda + \lambda'(b - Ax^*_\lambda) \\
&\geq cx^*_{\lambda'} + \lambda'(b - Ax^*_{\lambda'}) \quad (\text{since } x^*_{\lambda'} \text{ minimizes } cx + \lambda'(b - Ax)) \\
&= v(SLR(\lambda')),
\end{aligned}
$$

This proves that the function in $SLR(\lambda)$ is non-decreasing. This argument also proves the last statement because it implies that $v(SLR(\lambda')) \leq cx^*_\lambda + \lambda'(b - Ax^*_\lambda)$, then $v(SLR(\lambda')) - v(SLR(\lambda)) \leq cx^*_\lambda + \lambda'(b - Ax^*_\lambda) - v(SLR(\lambda))$ and therefore $v(SLR(\lambda')) - v(SLR(\lambda)) \leq (\lambda' - \lambda)(b - Ax^*_\lambda)$.

□

Because $SLR(\lambda)$ is concave and non-decreasing, then there is a set $[\lambda^*, +\infty)$ where with any of its elements we met the same optimal solution of $SLR(\lambda)$.

More details about semi-lagrangean relaxation can be seen in Beltrán et al. (2006, 2012).

4.3 A Lagrangean Relaxation for SPLPO

In this section we consider a Lagrangean Relaxation for the SPLPO. After trying different combinations of constraints to relax, we decided to relax (4.1) and (4.3) since as it will be shown later our first result shows that the proposed model is easy to solve. This will be particularly useful for the method that we will use to solve the associated Lagrangean dual problem.

If we relax constraints (4.1) and (4.3), then they are moved to the objective function with penalty coefficients (multipliers) when they are violated, thus obtaining the following Lagrangean relaxation problem $LR(\mu, \lambda)$:

$$
\begin{aligned}
&\min_{(x,y)} \sum_i \sum_j c_{ij} x_{ij} + \sum_j f_j y_j + \sum_i \mu_i \left(1 - \sum_j x_{ij}\right) + \sum_i \sum_j \lambda_{ij} \left[y_j - \sum_{k \in \overline{W_{ij}}} x_{ik}\right] \\
&= \min_{(x,y)} \sum_i \sum_j (c_{ij} - \mu_i) x_{ij} - \sum_i \sum_j \lambda_{ij} \sum_{k \in \overline{W_{ij}}} x_{ik} + \sum_j \left(f_j + \sum_i \lambda_{ij}\right) y_j + \sum_i \mu_i
\end{aligned}
$$

subject to: (4.2), (4.4) and (4.5).

The multiplier vectors μ and λ in $LR(\mu, \lambda)$ are unrestricted in sign and nonnegative, respectively.

Let $F(P)$ be the set of feasible solutions of problem P. Then for all $(x, y) \in F(\text{SPLPO})$ the objective function of $LR(\mu, \lambda)$ evaluated in (x, y) is always less than or equal to the objective function of P evaluated in (x, y). Therefore $v(LR(\mu, \lambda)) \leq v(\text{SPLPO})$.

In order to obtain the best lower bound for SPLPO, we need to solve the following Lagrangean dual problem:

$$
LD_{\mu\lambda} = \max_{\mu \in \mathbb{R}, \lambda \geq 0} LR(\mu, \lambda).
$$

Suppose that each customer i ranks the different potential facilities j with a number $p_{ij} \in \{1, \ldots, n\}$ with 1 and n the most and the least preferred, respectively. Since each multiplier λ_{ij} in a term of $\sum_i \sum_j \lambda_{ij} \sum_{k \in \overline{W_{ij}}} x_{ik}$ in $LR(\mu, \lambda)$ will be multiplied by a sum of p_{ij} variables x_{ik} with $k \geq_i j$, then each x_{ij} will be multiplied by a sum of $(n - p_{ij} + 1)$ values λ_{ik} with $k \leq_i j$. Therefore:

$$\sum_i \sum_j \lambda_{ij} \sum_{k \in \overline{W_{ij}}} x_{ik} = \sum_i \sum_j \left(\sum_{\substack{k \in W'_{ij} \\ |W'_{ij}| = n - p_{ij} + 1}} \lambda_{ik} \right) x_{ij},$$

and $LR(\mu, \lambda)$ can be rewritten as:

$$LR(\mu, \lambda) = \min_{(x,y)} \sum_i \sum_j \left(c_{ij} - \mu_i - \sum_{k \in W'_{ij}} \lambda_{ik} \right) x_{ij} + \sum_j \left(f_j + \sum_i \lambda_{ij} \right) y_j + \sum_i \mu_i,$$

subject to: (4.2), (4.4) and (4.5).

As in Cornuéjols et al. (1977), this Lagrangean problem is easy to solve analytically for fixed vectors μ and λ. If we define Λ_{ij} as:

$$\Lambda_{ij} = \sum_{k \in W'_{ij}} \lambda_{ik},$$

then we have the following result:

Theorem 14. *An optimal solution for $LR(\mu, \lambda)$ can be obtained as follows:*

$$y_j = \begin{cases} 1, & \text{if } \sum_i \min(0, c_{ij} - \mu_i - \Lambda_{ij}) + (f_j + \sum_i \lambda_{ij}) < 0, \\ 0, & \text{otherwise.} \end{cases}$$

and,

$$x_{ij} = \begin{cases} 1, & \text{if } y_j = 1 \text{ and } (c_{ij} - \mu_i - \Lambda_{ij}) < 0, \\ 0, & \text{otherwise.} \end{cases}$$

Proof. We have that:

$$\begin{aligned} LR(\mu, \lambda) &= \min_{(x,y)} \sum_i \sum_j \left(c_{ij} - \mu_i - \Lambda_{ij} \right) x_{ij} + \sum_j \left(f_j + \sum_i \lambda_{ij} \right) y_j + \sum_i \mu_i, \\ &= \min_{(x,y)} \sum_j \left[\sum_i \left(c_{ij} - \mu_i - \Lambda_{ij} \right) x_{ij} + \left(f_j + \sum_i \lambda_{ij} \right) y_j \right] + \sum_i \mu_i, \end{aligned}$$

subject to: (4.2), (4.4) and (4.5).

As a consequence of constraints (4.2) and for fixed vectors μ and λ, the optimal values for x_{ij} will be $x_{ij} = 1$ if $y_j = 1$ and $c_{ij} - \mu_i - \Lambda_{ij} < 0$, otherwise $x_{ij} = 0$. Then, if we define $\rho_j(\mu, \lambda) = \sum_i \min(0, c_{ij} - \mu_i - \Lambda_{ij}) + (f_j + \sum_i \lambda_{ij})$, then the optimal vector y can be obtained by solving the following minimization problem:

$$\min \sum_j \rho_j(\mu, \lambda) y_j,$$

subject to: $y_j \in \{0, 1\}$.

The solution to this problem is straightforward. □

An issue with this model is that $LR(\mu, \lambda)$ has the integrality property, that is, its optimal value is equal to the standard linear relaxation LP(SPLPO). Furthermore, the values obtained of the function $LR(\mu, \lambda)$ during the search of the solution for $LD_{\mu\lambda}$ are infeasible to the original problem SPLPO. So, as an alternative, we propose to use the solution of this problem as a starting point for another procedure that allows us to find feasible solutions to SPLPO.

4.3.1 Subgradient Method for the Lagrangean Dual $LD_{\mu\lambda}$

The subgradient method was originally proposed by Held and Karp (1971) and validated by Held et al. (1974). Given multipliers λ and μ, this method tries to optimize LD by taking steps along a subgradient of $LR(\mu, \lambda)$. A sketch of the whole procedure is given in Algorithm 4.3.1.

Algorithm 4.3.1 Subgradient method (SG) for SPLPO.

Let $LD = max_{(\mu,\lambda)} LR(\mu, \lambda)$ with $\mu \in \mathbb{R}$ and $\lambda \geq 0$.

Step 1. **(Initialization)**. Let LR_{AIM} by a heuristic method. Set $\beta = 2$. Let k be an integer number. Let q be a number in $[0, 1]$. Let $[\mu^0, \lambda^0]$ be a starting point.

Step 2. **(Obtaining values $x^0_{ij}, \forall i, j$ and $y^0_j, \forall j$)**. Find $LR^0_{best} = LR(\mu^0, \lambda^0)$. Set $iter = 0$.

Step 3. **(Finding a subgradient)**. Find a subgradient s^{iter} for $LR(\mu^{iter}, \lambda^{iter})$.

Step 4. **(Stop criterion)**. If $s^{iter} = 0$, STOP and $[\mu^{iter}, \lambda^{iter}]$ is optimal.
Otherwise, go to Step 5.

Step 5. **(Step size)**. Let $\alpha^{iter} = \beta \frac{LR_{AIM} - LR(\mu^{iter}, \lambda^{iter})}{\|s^{iter}\|_2^2}$.
If $LR(\mu^{iter}, \lambda^{iter})$ does not improve for k consecutive iterations, set $\beta = \beta * q$.

Step 6. **(Updating multipliers)** Set
$[\mu^{iter+1}, \lambda^{iter+1}] = [(\mu^{iter} + \alpha^{iter} s^{iter}_\mu), max(0, \lambda^{iter} + \alpha^{iter} s^{iter}_\lambda)]$.

Step 7. **(Updating incumbent)**. Let
$LR^{iter+1}_{best} = max\{LR^{iter}_{best}, LR(\mu^{iter+1}, \lambda^{iter+1})\}$. Update $iter = iter + 1$.

Step 8. **(Stop criterion)**. If $iter = MAXiter$, STOP. Otherwise go to Step 3.

There are other methods to solve the lagrangean dual (see Guignard (2003)). However, all of them must be adapted to the characteristics of a specific problem to make them more efficient. For example, in Erlenkotter (1978) is presented a dual-based procedure for the Simple Plant Location Problem (without order) that takes advantage of its linear programming dual formulation and with an ascent and adjustment procedure frequently produces optimal dual solutions that often correspond to optimal primal solutions. It is probably that for the SPLPO case this is not possible due to the inclusion of the customers' preference ordering constraints. As can be seen in Steps 2 and 5, at each iteration, we need to solve an instance of $LR(\mu, \lambda)$. Each of them is easy to solve, as shown in Theorem 14, so we decided to take advantage of this

property and use the subgradient method. The computation of the vector of subgradients is also easy:

$$s = \begin{bmatrix} 1 - \sum_{j \in J} x_{ij}; \forall i \\ y_j - \sum_{k \in \overline{W}_{ij}} x_{ik}; \forall i, j \end{bmatrix} = \begin{bmatrix} s_\mu \\ s_\lambda \end{bmatrix} \in \mathbb{R}^{m(n+1)},$$

is a subgradient for $LR(\mu, \lambda)$. If this vector is 0, then the procedure ends.

In our computational experiments, that will be showed later, an upper bound for LR_{AIM} in Step 5 is found using a simple heuristic. See Algorithm 4.3.2. First, it opens a facility that supplies all customers with the lowest operating cost and it is removed from the set $J' = J$. Then, for each unopened one, each customer compares and chooses the most preferred facility between it and its previously assigned supplier. The new cost is saved. The new open facility is the one with the lowest operating cost. It is removed from J'. This is repeated until $J' = \{\}$.

We also tried another heuristic. It is basically the same that for Hc with a different Step 4 which allows the algorithm to stop earlier. We name it Hs (see Algorithm 4.3.3) and its results will be reported later.

Algorithm 4.3.2 Heuristic to find an upper bound for SPLPO (Hc).

Let $G(I \times J, E = \{\})$ be a bipartite graph.

Step 1. Find $j_0 \in J$ such that $\sum_i c_{ij_0} = min\{\sum_i c_{i1}, \ldots, \sum_i c_{in}\}$.
Set $J' = J \setminus \{j_0\}$, $E = \{(i, j_0)\}$ and $TC_{prev} = \sum_i c_{ij_0}$.

Step 2. For all $j \in J'$, do:
Set $E_j = \{\}$.
For all $i \in I$, do:
Find k_{pref} such that $p_{ik_{pref}} = min[\{p_{ij}\} \cup \{p_{ik} \mid (i,k) \in E\}]$.
Set $E_j = E_j \cup \{(i, k_{pref})\}$.
Compute $TC_j = \sum_{(i,j) \in E_j} c_{ij}$.

Step 3. Find j_0 such that $TC_{j_0} = min\{TC_j \mid j = 1, \ldots, n\}$. Set $J' = J' \setminus \{j_0\}$ and $E = E_{j_0}$.

Step 4. If $J' = \{\}$, STOP. Otherwise go to Step 2.

Algorithm 4.3.3 Hs.

Step 1..3. Same as Hc.

Step 4. If $CT_{j_0} \geq CT_{prev}$ then STOP. Otherwise, set $CT_{prev} = CT_{j_0}$ and go to Step 2.

It is possible, due to an overestimation of LR_{AIM}, that the function $LR(\mu, \lambda)$ does not improve for many iterations. This can be overcome by setting the parameter β to a fixed value (for example, 2) and reducing it slowly.

In Step 6, we need vector λ to be nonnegative, therefore in the nonnegative orthant, i.e., $[\lambda_i]^+ = max\{0, \lambda_i\}$, for all of its components. μ must remain unchanged as it is an unrestricted vector.

Further details can be found in (Guignard, 2003; Conforti et al., 2014) and Poljak (1967) for the step size in Step 5 in Algorithm 4.3.1.

4.4 A Semi-Lagrangean Relaxation for SPLPO

Now we use the technique of the Semi-Lagrangean relaxation, as this leads to close the duality gap. The equality constraints (4.1) have been split into two inequalities $\sum_{j\in J} x_{ij} \leq 1$ and $\sum_{j\in J} x_{ij} \geq 1$ to obtain the following model:

$$\begin{aligned} SLR(\gamma) &= \min_{(x,y)} \sum_i \sum_j c_{ij}x_{ij} + \sum_j f_j y_j + \sum_i \gamma_i \left(1 - \sum_j x_{ij}\right), \\ &= \min_{(x,y)} \sum_i \sum_j (c_{ij} - \gamma_i)\, x_{ij} + \sum_j f_j y_j + \sum_i \gamma_i, \\ &= \min_{(x,y)} \sum_j \left(\sum_i (c_{ij} - \gamma_i)\, x_{ij} + f_j y_j\right) + \sum_i \gamma_i, \end{aligned}$$

subject to: (4.2), (4.3), (4.4), (4.5), and $\sum_{j\in J} x_{ij} \leq 1$. Every component of the multipliers vector γ is nonnegative.

As mentioned in Section 4.2.1, the objective function is concave and non-decreasing in its domain. Therefore, its semi-Lagrangean dual problem:

$$SLD_\gamma = \max_{\gamma\geq 0} SLR(\gamma),$$

can be solved using an ascent method. Also, as pointed out, there is a set $Q = [\gamma^*, +\infty)$ such that, for any $q \in Q$ the optimum of $SLR(\gamma)$ is met.

For the SPLP (no preferences), Beltrán et al. (2012) proved that there is a closed interval where the search of the multipliers could be done. Following the same idea, we provide the next two results for the SPLPO:

Theorem 15. *Let $cp_i = max_j\{c_{ij} + f_j\}$ and let $cp = (cp_1, \ldots, cp_m)$ be the maximum of the costs for each customer i associated to each facility j and the vector of these costs, respectively. If $\gamma \geq cp$, then $\gamma \in Q$.*

Proof. As we know, the semi-Lagrangean relaxation closes the duality gap if for all i, $\sum_{j\in J} x_{ij} = 1$. Assume that $SLR(\gamma) < \infty$, otherwise the proposition is trivially true. By hypothesis, $cp_i - \gamma_i \leq 0$ for all $i \in I$. If we choose j' such that $cp_i = c_{ij'} + f_{j'}$, it turns out in $(c_{ij'} - \gamma_i) + f_{j'} \leq 0$, and this inequality is true for any j since j' gives the maximum among all $c_{ij} + f_j$. Therefore, the event $\sum_{j\in J} x_{ij} = 0$ can not happen at an optimal solution, because it is always possible to set $x_{ij} = 1$ and $y_j = 1$ for all i and j meeting all the constraints. □

Theorem 16 (Beltrán et al. (2012))**.** *For each $i \in I$, let $c_i^1 \leq \ldots \leq c_i^n$ be the sorted costs c_{ij}. If $\gamma < c^1$ then $\gamma \notin Q$.*

Proof. By hypothesis $c_i^1 - \gamma_i > 0$, then $c_i^j - \gamma_i > 0$ for all $j \in J$. In that case, $x_{ij}^* = 0$ for all $j \in J$ in any optimal solution x_γ^*. Therefore, $\sum_{j\in J} x_{ij} = 1$ can not happen at an optimal solution and $\gamma \notin Q$. □

The result of Theorem 15 is weaker than the obtained by Beltrán et al. (2012) for SPLP in the following sense. They choose cp_i equal to $min_j\{c_{ij} + f_j\}$, but with this, there is no guarantee that the preference constraints hold. We can set $x_{ij'}^* = 1$ and $y_{j'} = 1$ for the particular j' such that $min_j\{c_{ij} + f_j\} = c_{ij'} + f_{j'}$ but not necessarily for all j, as in this way, are not available all

the possible combinations of x's and y's that consider all customers preferences. Therefore, we are taking the risk of obtaining a non optimal solution.

The interval of search for γ components is then:

$$B = \{\gamma \mid c^1 < \gamma \leq cp\}. \tag{4.8}$$

Using the sorted costs $c_i^1 \leq \ldots \leq c_i^n$, each component γ_i of γ can be either in an interval of the form $I_j = (c_i^j, c_i^{j+1}]$ where $j \in \{1, \ldots, m-1\}$ or out of it. For the first case, there are infinite values of γ_i that can belong to a single interval $(c_i^j, c_i^{j+1}]$, but each of them has the same effect on the optimal value of $SLR(\gamma)$. This is because going from an interval I_j to the next I_{j+1} could imply a change in the choice of the arc (i,j) to the arc $(i,j+1)$ in the bipartite graph $G_\gamma = (I \times J, E = \{(i,j) \mid x_{ij} = 1 \text{ in the solution of } SLR(\gamma)\})$, which means a change in the solution. Hence, we just need a single γ_i representative of the intervals. As γ goes to infinity all combined costs $(c_{ij} - \gamma_i) + f_j$ become negative, and hence the semi-Lagrangean relaxation problem can be as difficult to solve as the original SPLPO problem. Then, it is always convenient to choose a $\gamma_i \in I_i$ as smaller as possible, that is, at an epsilon ϵ distance from the lower bound of an interval. Also, it is easy to check that c_i^n is always less than cp_i if $f_j \geq 0$. These ideas will be applied in the ascent method used later.

4.4.1 Dual Ascent Method for the Semi-Lagrangean Dual SLD_γ

In general, a dual ascent algorithm modifies a current value of multipliers in order to achieve a steady increase in $SLR(\gamma)$, see Bilde and Krarup (1977). In our case this is possible due to the aforementioned properties. The method we used is explained under Algorithm 4.4.1.

Algorithm 4.4.1 Dual ascent method (DA) for SLD_γ.

Let $c_i^1 \leq \ldots \leq c_i^n$ be the sorted costs c_{ij} and let γ^0 be an initial vector γ.

Step 1. **(Initialization)**. Set $\epsilon > 0$, $iter = 0$. If $k > n$, $c_i^k = cp_i$.
For all $i \in I$ do:
$cp_i = max_j\{c_{ij} + f_j\}$,
If $\gamma_i^0 < c_i^1$, then $\gamma_i^0 = c_i^1 + \epsilon$,
Else,
If $\gamma_i^0 > c_i^n$, then $\gamma_i^0 = c_i^n$,
Else,
find a j_i such that γ_i belongs to $(c_i^{j_i}, c_i^{j_i+1}]$, and set $\gamma_i^0 = c_i^{j_i} + \epsilon$.

Step 2. **(Obtaining values x_{ij}^{iter} and y_j^{iter})**. Solve $SLR(\gamma^{iter})$.

Step 3. **(Finding a subgradient)**. Find a subgradient s^{iter} of $SLR(\gamma^{iter})$.

Step 4. **(Stop criterion)**. If $s^{iter} = 0$, STOP and γ^{iter} is optimal. Otherwise, go to Step 5.

Step 5. **(Updating multipliers)** For each i such that $s_i^{iter} = 1$, $j_i = j_i + 1$ and $\gamma_i^{iter+1} = min\{c_i^{j_i} + \epsilon, cp_i\}$. Set $iter = iter + 1$. Go to step 2.

In Step 1, the algorithm gives an appropriate position of the terms of an initial multipliers vector in the intervals I_j. In Step 2, the procedure needs to solve a sequence of problems $SLR(\gamma)$, but due to the preference constraints, they are not as easy to solve as in the Lagrangean relaxation case $LR(\mu, \lambda)$ proposed before. However, it is possible to set some variables x_{ij} equal to 0 in advance, in order to make it easier. Every x_{ij} must be 0 if $c_{ij} - \gamma_i > 0$.

A subgradient for $SLR(\gamma)$ is computed in Step 3 as:

$$s = \left[1 - \sum_{j \in J} x_{ij}; \forall i \right] = [s_\gamma] \in \mathbb{R}^m,$$

where each component of s belongs to $\{0, 1\}$ as the $\sum_{j \in J} x_{ij} \leq 1$ must be satisfied.

In Step 4, a stop criterion is given. Finally, the multipliers are updated (increased) by jumping from the current to the next I_j interval for each component i in γ.

4.5 Speeding Up the Search for the Optimal Solution

After a certain number of iterations of DA that has been fixed beforehand, we apply a variable fixing heuristic VFH that takes, among all $y_j = 1$, a percentage of them by sorting this set of y_j's by a determined criterion. Then, it sets all the selected y_j equal to 1. Finally the subproblem is solved. The method is described under Algorithm 4.5.1.

Algorithm 4.5.1 Variable fixing heuristic (VFH) to speed up DA.

Let SLR_γ be an instance of DA after k iterations.
Let Y_γ be a set of y_j's equal to 1 at instance SLR_γ.

Step 1. Set $ps \in [0, 1] \in \mathbb{R}$. Sort the elements of Y_γ such that $y_j \prec y_k$ if $\sum_i (c_{ij} + f_j) < \sum_i (c_{ik} + f_k)$.

Step 2. Choose the first $ps \times 100\%$ smallest elements from the sorted Y_γ to form the set Y_γ^{subset}.

Step 3. Set, $y_j = 1$ for all $y_j \in Y_\gamma^{subset}$.

Step 4. Solve the original $SPLPO$.

Step 1 shows the criterion that we used to sort variables y_j. We also tried to sort it by $y_j \prec y_k$ if $\sum_i p_{ij} < \sum_i p_{ik}$, where the p's are the preferences. We obtained similar results, but the option proposed in Algorithm 4.5.1 was slightly better.

Another important issue is the starting point for the DA. We have tried two different starting γ vectors, $\gamma = 0$ and γ equal to μ, which is one of the multipliers that can be found using the subgradient method SG for $LD_{\mu\lambda}$ after a certain number of iterations. This μ is the penalization associated with the relaxed family of constraints (4.1). We obtained better results with the second approach. Furthermore, each sub-problem $LR(\mu, \lambda)$ to be solved in the optimization of $LD_{\mu\lambda}$ is easy, see Theorem 14. For SG we used $\mu_i^0 = min_j\{c_{ij} + f_j\}$ for all $i \in I$ and $\lambda_{ij} = 0$ for all $i \in I$ and $j \in J$.

The whole accelerated dual ascent procedure is presented in Algorithm 4.5.2.

Algorithm 4.5.2 Accelerated dual ascent algorithm. (ADA).

Step 1. Set $\mu_i^0 = min_j\{c_{ij} + f_j\}$ for all $i \in I$ and $\lambda_{ij} = 0$ for all $i \in$ and $j \in J$.

Step 2. Run SG(μ^0, λ^0) during *sg_iter* iterations. Find the iteration *best_sg_iter* where is the best value of $LR(\mu, \lambda)$. Set $\gamma^0 = \mu^{best\_sg\_iter}$.

Step 3. During *da_iter* iterations, run DA(γ^0).

Step 4. During vfh_iter run DA. Get Y_γ and run VFH. Save the solution.

Step 5. Find the best solution in the history and get best values for x_{ij} and y_j.

4.6 Computational Results

In this section we present the computational results obtained after having applied the algorithms shown before. The experiments have been carried out on a PC with Intel® Xeon® 3.40 GHz processor and 16 Gb of RAM under a Windows® 7 operative system. All the procedures and algorithms have been written using the version 4.0.3 of the Mosel Xpress language and when a MIP problem needed to be solved we used FICO Xpress® version 8.0.

The instances were, at first, taken from Cánovas et al. (2006), which are based on the Beasley's OR-Library (see Beasley, 1990). These are:

131, 132, 133, 134: $m = 50$ and $n = 50$,
a75_50, b75_50, c75_50: $m = 75$ and $n = 50$,
a100_75, b100_75, c100_75: $m = 100$ and $n = 75$.

Additional data sets were generated using the same algorithm proposed in Cánovas et al. (2006):
a125_100, b125_100, c125_100: $m = 125$ and $n = 100$,
a150_100, b150_100, c150_100: $m = 150$ and $n = 100$.

The headers of the tables have the following meanings:

- Prob: Name of the problem.
- Opt: Optimal objective function value of the problem P.
- y_j: Number of opened facilities.
- bestUB: Best upper bound.
- GAP_o (%): bestUB-Opt ($\frac{\text{bestUB-Opt}}{\text{Opt}} \times 100\%$). The absolute and relative gap between the optimal and the value of an algorithm.
- GAP_LP: $\frac{\text{LP(P)-SG}}{\text{LP(P)}} \times 100\%$. The relative gap between the linear relaxation and the lower bound of the Lagrangian relaxation.
- LP(P): Linear relaxation value for problem P.
- SG: Best value using the subgradient method for $LD_{\mu\lambda}$.
- iter (itbsol.): Number of iterations (iteration where best solution was found).
- t (Tt): CPU time in seconds (Total time).
- sol10%: First solution found that is within 10% or less of the optimal solution.
- UB (LB): Upper bound (Lower bound).

First, we show in Table 4.1 the results for Algorithms Hs and Hc. As can be seen, we obtain better results with the second one. It finds the optimal solution for 6 of the first 16 problems and in the rest of the instances it is not too far from the optimal with an average gap of 4.65%. All runs performed took less than 1 second whereas Xpress needs between 3 and 12 to solve these instances to optimality. The value obtained by Hc was used as an upper bound for the SG method, Algorithm 4.3.1.

Table 4.1: Comparison between upper bounds heuristics Hs and Hc.

			Hs				Hc			
Prob	**Opt**	**y_j**	**bestUB**	**GAP$_o$**	**GAP$_o$%**	**y_j**	**bestUB**	**GAP$_o$**	**GAP$_o$%**	**y_j**
131_1	1001440	6	1001440	0	0.00%	6	1001440	0	0.00%	6
131_2	982517	9	1003100	20583	2.09%	5	982517	0	0.00%	9
131_3	1039853	10	1248143	208290	20.03%	1	1139012	99159	9.54%	9
131_4	1028447	9	1248143	219696	21.36%	1	1049791	21344	2.08%	7
132_1	1122750	8	1248143	125393	11.17%	1	1239961	117211	10.44%	7
132_2	1157722	9	1248143	90421	7.81%	1	1199057	41335	3.57%	9
132_3	1146301	6	1248143	101842	8.88%	1	1146301	0	0.00%	6
132_4	1036779	5	1248143	211364	20.39%	1	1036779	0	0.00%	5
133_1	1103272	7	1248143	144871	13.13%	1	1103272	0	0.00%	7
133_2	1035443	5	1248143	212700	20.54%	1	1100713	65270	6.30%	7
133_3	1171331	6	1248143	76812	6.56%	1	1208198	36867	3.15%	4
133_4	1083636	9	1248143	164507	15.18%	1	1090582	6946	0.64%	9
134_1	1179639	4	1248143	68504	5.81%	1	1179639	0	0.00%	4
134_2	1121633	7	1248143	126510	11.28%	1	1205809	84176	7.50%	5
134_3	1171409	6	1248143	76734	6.55%	1	1173693	2284	0.19%	7
134_4	1210465	3	1248143	37678	3.11%	1	1248143	37678	3.11%	1
a75_50_1	1661269	7	1787955	126686	7.63%	1	1787955	126686	7.63%	1
a75_50_2	1632907	6	1784848	151941	9.30%	2	1779576	146669	8.98%	4
a75_50_3	1632213	7	1738404	106191	6.51%	3	1738404	106191	6.51%	3
a75_50_4	1585028	5	1787955	202927	12.80%	1	1709978	124950	7.88%	5
b75_50_1	1252804	8	1374685	121881	9.73%	8	1343201	90397	7.22%	10
b75_50_2	1337446	9	1403629	66183	4.95%	5	1403629	66183	4.95%	5
b75_50_3	1249750	9	1368788	119038	9.52%	8	1368788	119038	9.52%	8
b75_50_4	1217508	9	1348203	130695	10.73%	10	1348203	130695	10.73%	10
c75_50_1	1310193	11	1390321	80128	6.12%	8	1375386	65193	4.98%	11
c75_50_2	1244255	10	1316595	72340	5.81%	11	1316595	72340	5.81%	11
c75_50_3	1201706	12	1386817	185111	15.40%	13	1386817	185111	15.40%	13
c75_50_4	1334782	11	1440836	106054	7.95%	5	1440836	106054	7.95%	5
a100_75_1	2286397	4	2476632	190235	8.32%	1	2459349	172952	7.56%	4
a100_75_2	2463187	3	2476632	13445	0.55%	1	2476632	13445	0.55%	1
a100_75_3	2415836	3	2476632	60796	2.52%	1	2467003	51167	2.12%	3
a100_75_4	2380150	4	2476632	96482	4.05%	1	2476632	96482	4.05%	1
b100_75_1	1950231	8	2061201	110970	5.69%	7	2061201	110970	5.69%	7
b100_75_2	2023097	8	2389395	366298	18.11%	1	2180286	157189	7.77%	8
b100_75_3	2062595	8	2133724	71129	3.45%	8	2133724	71129	3.45%	8
b100_75_4	1865323	9	1994265	128942	6.91%	7	1994265	128942	6.91%	7
c100_75_1	1843620	6	2107973	264353	14.34%	5	2090221	246601	13.38%	7
c100_75_2	1808867	11	2025331	216464	11.97%	9	2005071	196204	10.85%	13
c100_75_3	1820587	8	2019651	199064	10.93%	7	2019651	199064	10.93%	7
c100_75_4	1839007	9	2046525	207518	11.28%	8	2046525	207518	11.28%	8

Table 4.2 shows the performance of the subgradient method SG (Algorithm 4.3.1) over the first 40 data sets. The algorithm was stopped when the parameter β was equal to 0 with a starting value of 2, and it was decreasing linearly at a rate of 0.005 if a solution did not change after 30 iterations. Due to the behaviour of the SG, we also show the results for the first solution within a 10% from LP(P). This shows that SG method can be stopped earlier to obtain similar objective function values, see Figure 4.1.

Table 4.2: Computational results of subgradient method.

		Until $\beta = 0$					**First solution $< 10\%$**			
Prob	**LP(P)**	**SG**	**itbsol**	**iter**	**t**	**GAP_{LP}**	**sol**10%	**iter**	**t**	**GAP_{LP}**
131_1	925492	881876	1474	1500	19	4.71%	833518	613	8	9.90%
131_2	925195	872136	1450	1500	19	5.73%	833747	695	9	9.90%
131_3	955447	929854	1488	1500	20	2.68%	860178	636	8	10.00%
131_4	933025	903486	1462	1500	19	3.17%	841581	580	7	9.80%
132_1	1007417	981624	1500	1500	19	2.56%	907222	479	6	9.90%
132_2	990513	957530	1389	1425	18	3.33%	897874	445	6	9.40%
132_3	1009054	974902	1500	1500	19	3.38%	911244	658	8	9.70%
132_4	966305	910344	1388	1434	18	5.79%	869850	604	8	10.00%
133_1	998199	958109	1395	1498	19	4.02%	898706	593	8	10.00%
133_2	971719	947626	1443	1500	19	2.48%	882263	423	5	9.50%
133_3	1023593	982907	1473	1500	19	3.97%	927636	565	7	9.40%
133_4	1001253	948311	1489	1500	19	5.29%	902108	634	8	9.90%
134_1	1226933	1013753	1192	1226	16	2.21%	933610	375	5	9.90%
134_2	1041770	998540	1165	1239	16	4.15%	942697	530	7	9.50%
134_3	1023070	994435	1155	1286	16	2.80%	921339	415	5	9.90%
134_4	1050134	1003356	980	1026	13	4.45%	945217	439	6	10.00%
a75_50_1	1201542	1169011	732	850	17	2.71%	1096795	106	2	9.40%
a75_50_2	1188359	1158500	829	896	18	2.51%	1076464	107	2	9.40%
a75_50_3	1189183	1167203	936	999	20	1.85%	1073799	157	3	9.70%
a75_50_4	1200068	1175405	854	937	19	2.06%	1094899	106	2	8.80%
b75_50_1	900617	879687	1118	1182	24	2.32%	812083	334	7	9.80%
b75_50_2	922048	896384	1111	1211	25	2.78%	830962	381	8	9.90%
b75_50_3	897073	882132	1208	1271	25	1.67%	808366	326	7	9.90%
b75_50_4	885392	856157	784	925	19	3.30%	801482	281	6	9.50%
c75_50_1	892837	879074	845	876	18	1.54%	804437	304	6	9.90%
c75_50_2	882933	870772	1120	1175	23	1.38%	797745	317	6	9.60%
c75_50_3	859591	852325	1013	1050	22	0.85%	786773	347	7	8.50%
c75_50_4	928064	891598	924	1084	22	3.93%	837696	415	8	9.70%
a100_75_1	1800032	1725539	635	721	41	4.14%	1626622	99	6	9.60%
a100_75_2	1793377	1723484	679	728	42	3.90%	1618944	99	6	9.70%
a100_75_3	1788395	1724444	664	745	42	3.58%	1632328	103	6	8.70%
a100_75_4	1795298	1702235	482	678	38	5.18%	1620377	100	6	9.70%
b100_75_1	1330138	1289189	1010	1116	63	3.08%	1198182	187	11	9.90%
b100_75_2	1353824	1284338	652	691	39	5.13%	1225149	373	21	9.50%
b100_75_3	1351603	1306286	987	1044	59	3.35%	1217952	270	15	9.90%
b100_75_4	1332525	1258497	432	679	38	5.56%	1201093	157	6	9.90%
c100_75_1	1250876	1168284	510	710	40	6.60%	1127235	303	17	9.90%
c100_75_2	1239182	1154640	533	738	42	6.82%	1119427	424	24	9.70%
c100_75_3	1232462	1173522	670	804	46	4.78%	1115951	270	15	9.50%
c100_75_4	1243861	1194966	895	942	54	3.93%	1121515	356	20	9.80%

After the time reported, SG did not obtain the value $LP(P)$ for any of the problems. However, as mentioned before, the SG will be useful to find initial multipliers for the dual ascent procedure DA.

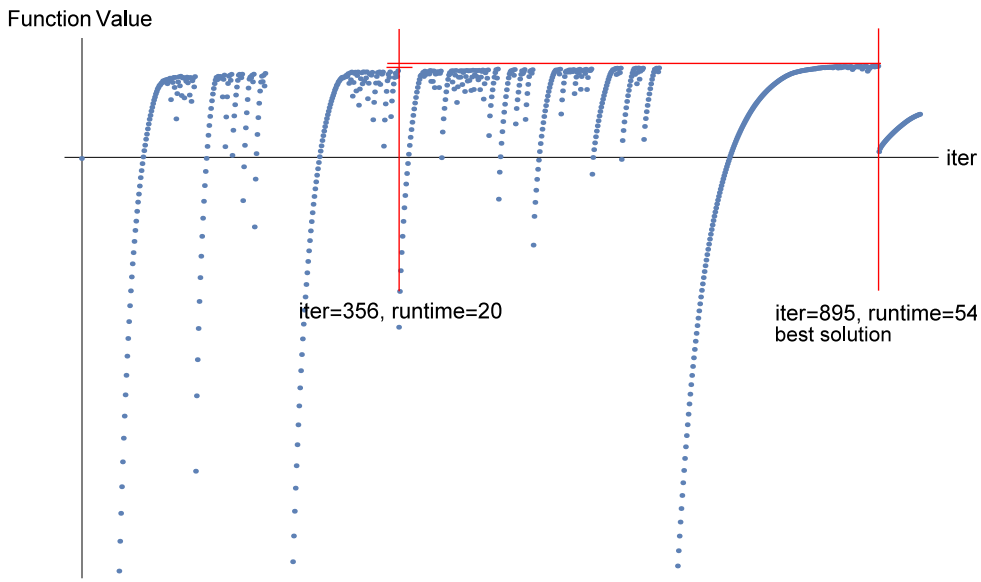


Figure 4.1: An example of SG after 1000 iterations.

Finally, in Table 4.3 we compare the results obtained by applying the ADA algorithm to the four largest groups of instances with the optimal solution obtained by Xpress. The table shows times and bounds reached by the subgradient method (**SG**) and by the dual ascent with the variable fixing heuristic procedure (**DA with VHF**). The total time used by the complete procedure is given in the column **Tt** and in **imp t** its improvement with respect to the time needed by Xpress to meet the optimal. The difference between the best upper bound obtained by ADA and the optimum is given as a percentage in $\mathbf{GAP_o}$.

The routine needs parameters sg_iter, da_iter and vfh_iter, and different values were used for the four groups of data sets tested. These were empirically set as:

a75_50, b75_50, c75_50: $sg\_iter = 50$, $da\_iter = 3$ and $fhv\_iter = 2$,

a100_75, b100_75, c100_75: $sg\_iter = 100$, $da\_iter = 7$ and $fhv\_iter = 2$,

a125_100, b125_100, c125_100: $sg\_iter = 170$, $da\_iter = 10$ and $fhv\_iter = 2$,

a150_100, b150_100, c150_100: $sg\_iter = 170$, $da\_iter = 12$ and $fhv\_iter = 2$,

and for all the instances $ps = 0.25$ (see Algorithm 4.5.1).

With the exception of the first group ($m = 75$, $n = 50$) and a few instances in the rest, the times obtained by ADA when it meets the optimal solution are considerably lower than those obtained by Xpress. It can be observed particularly in the last group, For example, the instance c_150_100_2 had a reduction of 95% on the total time. ADA is always close to the optimal values, in fact, the GAP average is only 0.43%.

Table 4.3: Computational results for ADA.

	Xpress				Hc			SG		DA with VFH				ADA		
Prob	**Optimal**	**LP**	**y_j**	**t**	**bestUB**	**y_j**	**t**	**LB**	**t**	**bestUB**	**LB**	**y_j**	**t**	**Tt**	**imp t**	**GAP$_o$%**
a75_50_1	1661269	1201542	7	16	1787955	1	0	832797	4	1662877	1290126	5	21	25	-51%	0.10%
a75_50_2	1632907	1188359	6	21	1779576	4	0	828462	4	1648421	1261845	6	28	31	-47%	0.95%
a75_50_3	1632213	1189183	7	18	1738404	3	0	783044	3	1632866	1286922	7	24	27	-51%	0.04%
a75_50_4	1585028	1200068	5	18	1709978	5	0	863178	3	1585028	1250104	5	28	31	-72%	0.00%
b75_50_1	1252804	900617	8	11	1343201	10	0	355739	2	1252804	951766	8	28	30	-165%	0.00%
b75_50_2	1337446	922048	9	20	1403629	5	0	376669	3	1337446	954023	9	35	38	-94%	0.00%
b75_50_3	1249750	897073	9	17	1368788	8	0	348636	3	1255947	975724	11	39	42	-141%	0.50%
b75_50_4	1217508	885392	9	12	1348203	10	0	414596	3	1222561	978717	10	35	38	-232%	0.42%
c75_50_1	1310193	892837	11	17	1375386	11	0	550106	4	1310193	972827	11	38	42	-151%	0.00%
c75_50_2	1244255	882933	10	11	1316595	11	0	533586	3	1244255	964720	10	36	39	-244%	0.00%
c75_50_3	1201706	859591	12	8	1386817	13	0	549787	4	1201706	934664	12	33	37	-341%	0.00%
c75_50_4	1334782	928064	11	21	1440836	5	0	581078	4	1356714	974828	12	41	45	-111%	1.64%
a100_75_1	2286397	1800032	4	61	2459349	4	1	1649771	16	2321812	1950826	4	74	91	-49%	1.55%
a100_75_2	2463187	1793377	3	131	2476632	4	1	1630225	16	2476632	1982594	3	106	122	7%	0.55%
a100_75_3	2415836	1788395	3	148	2467003	3	1	1634629	16	2415836	1970116	3	99	116	22%	0.00%
a100_75_4	2380150	1795298	4	124	2476632	3	1	1628551	17	2386981	1938879	4	99	116	7%	0.29%
b100_75_1	1950231	1330138	8	624	2061201	7	1	1092987	17	1984720	1511894	10	125	142	77%	1.77%
b100_75_2	2023097	1353824	8	727	2180286	8	1	1150334	18	2058495	1517533	11	143	161	78%	1.75%
b100_75_3	2062595	1351603	8	841	2133724	8	1	1165120	18	2062595	1536082	8	268	285	66%	0.00%
b100_75_4	1865323	1332525	9	546	1994265	7	1	1143446	15	1886598	1512185	7	141	155	72%	1.14%
c100_75_1	1843620	1250876	6	430	2090221	7	1	1077180	20	1843620	1456543	6	215	235	45%	0.00%
c100_75_2	1808867	1239182	11	396	2005071	13	1	1053868	19	1815373	1415172	9	171	190	52%	0.36%
c100_75_3	1820587	1232462	8	423	2019651	7	1	1054526	16	1820587	1405713	8	194	210	50%	0.00%
c100_75_4	1839007	1243861	9	583	2046525	8	1	1064170	21	1839007	1417402	9	227	248	57%	0.00%
a125_100_1	3041451	2392412	2	257	3070535	1	1	2235753	47	3070535	2539124	2	151	198	23%	0.96%
a125_100_2	3040248	2393448	2	302	3070535	1	1	2232859	48	3070535	2629291	2	201	249	18%	1.00%
a125_100_3	3055260	2362216	3	325	3070535	1	1	2227182	47	3070535	2593865	3	209	256	21%	0.50%
a125_100_4	3056428	2381167	2	334	3070535	1	1	2244017	54	3070535	2529404	2	208	262	22%	0.46%
b125_100_1	2640798	1794710	7	5297	2850664	5	1	1601985	58	2640798	2082726	7	989	1047	80%	0.00%
b125_100_2	2550592	1790869	7	2086	2808259	9	1	1600582	56	2568522	2046667	7	848	903	57%	0.70%
b125_100_3	2604906	1792133	4	4059	2782609	5	1	1607076	58	2604906	1977647	4	435	494	88%	0.00%
b125_100_4	2580595	1792581	7	2686	2778713	4	1	1587315	60	2637611	1975516	7	358	419	84%	2.21%

Continued on next page

Table 4.3: Computational results for ADA.

	Xpress				Hc			SG		DA with VFH				ADA		
Prob	**Optimal**	**LP**	**y_j**	**t**	**bestUB**	**y_j**	**t**	**LB**	**t**	**bestUB**	**LB**	**y_j**	**t**	**Tt**	**imp t**	**GAP$_o$%**
c125_100_1	2491714	1663455	10	7805	2669965	9	1	1491505	61	2491714	1953129	10	1685	1746	78%	0.00%
c125_100_2	2468480	1674102	9	6296	2674261	8	1	1487416	58	2518055	1957245	10	519	577	91%	2.01%
c125_100_3	2559381	1672005	8	11381	2685807	7	1	1496552	59	2559381	1957575	8	1177	1236	89%	0.00%
c125_100_4	2538550	1691148	8	6407	2683676	9	1	1503317	61	2561098	2016969	8	726	786	88%	0.89%
a150_100_1	3768087	2906284	1	371	3768087	1	2	2691219	57	3768087	3200250	3	338	395	-6%	0.00%
a150_100_2	3768087	2891681	1	457	3768087	1	1	2695660	56	3768087	3212545	2	329	384	16%	0.00%
a150_100_3	3741364	2882917	3	340	3768087	1	1	2691438	58	3741364	3185719	3	306	363	-7%	0.00%
a150_100_4	3768087	2911017	1	591	3768087	1	1	2697003	63	3768087	3241759	2	337	400	32%	0.00%
b150_100_1	3271859	2159537	4	15483	3487710	4	1	1916485	68	3271859	2651812	4	2357	2426	84%	0.00%
b150_100_2	3227987	2160753	5	9677	3457623	5	2	1935777	68	3227987	2546429	5	975	1043	89%	0.00%
b150_100_3	3150075	2148658	6	5986	3390435	8	1	1920876	69	3150075	2582861	6	1081	1150	81%	0.00%
b150_100_4	3342783	2190488	5	9596	3637438	8	1	1927555	69	3342783	2602414	5	2225	2294	76%	0.00%
c150_100_1	2979389	1988908	9	10008	3196861	5	1	1790894	71	2979389	2408302	9	1520	1591	84%	0.00%
c150_100_2	3109105	1985389	7	21734	3291990	6	2	1766696	72	3109105	2404613	7	949	1021	95%	0.00%
c150_100_3	2937767	1982388	8	10590	3079664	4	1	1770318	71	2954519	2379867	7	2369	2440	77%	0.57%
c150_100_4	3165327	1997014	10	65134	3189247	7	1	1775186	75	3176587	2424997	12	5842	5917	91%	0.36%

4.7 Conclusions

In this paper we have proposed a heuristic method to solve the SPLPO inspired by techniques introduced in (Cornuéjols et al., 1977; Beltrán et al., 2012). The assignment and VUBs constraints in SPLPO were relaxed to formulate a Lagrangean problem. We solved its dual with a subgradient method and used a vector of multipliers as a starting point of an ascent algorithm for the dual of a semi-Lagrangean formulation. Nevertheless, a better starting point should be the multipliers obtained by relaxing only the assignment constraints, the same family of constraints relaxed in our proposed SPLPO semi-Lagrangean problem. However, the sequence of linear subproblems that need to be solved in the subgradient method are not as easy to solve as those when our proposed relaxation is used. We used the variable fixing heuristic VFH because MIP problems in ADA are becoming increasingly difficult as the number of iterations grows. We have shown that the ADA algorithm works particularly well on large instances, but there must be a future discussion about the parameters settings.

Chapter 5

The Stochastic Simple Plant Location Problem with Partial Order

In this chapter we generalize the Simple Plant Location Problem with Order (SPLPO) by defining *partial order* and providing a linear model that considers this case. The partial order is a natural extension of SPLPO which takes into account the fact that some customers can have preference only over a subset of facilities that will provide them a service.

We also study how the uncertainty can be included in the model when the preferences are considered as random variables.

5.1 Introduction

Sometimes by taking certain problems to a realistic application it is possible to observe that certain unknown events are unveiled in the course of time, and these in turn help to solve others that depend on them. An example is the Simple Plant Location Problem (SPLP) whose solution can be seen as a two-stage process that answers two questions: what facility to open? and how will these facilities be distributed to meet customer demand?. The second question will be answered after the facilities (uncertain at the beginning) are known. The values of unknown variables in the problem will be unveil after an experiment, that considers a possible future state, is performed. Therefore, it is necessary to define different forms of that state (state of nature) that represent possibles scenarios in such a way that the effect of considering uncertainty is minimized. The problem described above is called Two-stage Stochastic SPLP problem (2S-SPLP).

The 2S-SPLP has been widely studied and the different methods of solution applied have been very efficient. As it is pointed out in Birge and Louveaux (2014), it is important to consider that "taking advantage of structure of the problem is especially beneficial in stochastic programs and is the focus of much of the algorithmic work in this area". "Structures" refers to any feature in the mathematical formulation that can be used to build a solution procedure. One of the most used method is called L-shaped, proposed by Laporte and Louveaux (1993). L-shaped is a branch and cut algorithm that makes use of the L shape that has the matrix of coefficients in the stochastic formulation of certain problems (SPLP is an example). This method also uses subproblems obtained by Benders decomposition (Benders, 1962). For details of this and other similar methods, see Kall and Mayer (2005) and Birge and Louveaux (2014). Although in principle Lagrangean relaxation methods can be applied to two-stage stochastic problems, as far as we know, these are not popular. However, a completed description of the different possibilities of its application can be found in the references given above.

A multiple-stage stochastic version for SPLP can be developed, as presented in Wollenweber (2008). In this paper the multiple stages (for the deterministic case of SPLP) are the consequence of considering the end-of-life of vehicles that will distribute the products from the facilities to the customers in the context of Supply Chain Management. However, this will not be considered for the problem we are interested in, the natural stochastic extension for Simple Plant Location Problem with Order (SPLPO). Therefore a two-state problem is more suitable in our case. In order to make the problem more realistic, we also study the possibility that the order of preference given by customers in SPLPO, which ranks the facilities that will serve them, is partial, i.e., a client would like to be served by any facility in a subset of the complete facility set. This is motivated by cases such as the following. In health services, service centres must be located in strategic locations that meet the immediate needs of individuals. Let's assume that there is a doctor for each centre. In some cases, due to the quality of service, empathy, price or distance, clients may prefer one doctor more than another. This defines a ranking (preference ordering) which will not necessarily be given for all them, since a patient could never want to be seen by a doctor in a particular health center.

In our proposed formulation for the two-stage stochastic version of SPLPO with partial order (2S-SPLPPO), along with the facilities to be opened, the preferences (partial or complete) will be considered as variables in the first stage.

Due to the promising results obtained in Chapter 4 by using Lagrangean and Semi-Lagrangean procedures, we have chosen to use these methods to solve the 2S-SPLPPO. As it will be seen later, the application of the ADA algorithm 4.5.2 to the 2S-SPLPPO to a group of problems derived from those given in Section 4.6 produced considerably good results when compared with the solutions given by a commercial optimization software (Xpress).

The rest of the chapter is as follows: In Section 5.2 we provide a basic background on stochastic programs. In Section 5.3 The Simple Plant Location Problem with Partial Order is defined and a formulation for this case is given in Section 5.4. Numerical experiments are carried out in Section 5.5 to check the impact that different scenarios of order have on the proposed model. Furthermore, we justify the use of ADA algorithm 4.5.2 to 2S-SPLPPO and apply it on some large stochastic instances.

5.2 Preliminaries

Basic background of events, random variables and probability is reviewed in this section. We also briefly look at two-stage stochastic programs.

5.2.1 Events, Random Variables and Probability

We consider an *experiment* as any process of observation or measurement (Freund et al., 2014), i.e., it is a checking process. All the experiment outcomes (results of the experiment) can be gathered in the set called *sample space* which is usually noted by Ω. The elements of Ω are called either simply *elements* or *sample points*. Ω can be classified for their number of elements. It can be finite, infinite countable (*discrete*) or not countable (*continuous*). In this thesis we are interested only in the discrete case, so that is the theory that will be summarized in this section.

An *event* E of Ω is a collection of sample points, i.e., a subset of a *sample space*, and therefore on them any set operations can be applied. Each event E is associated with a value $P(E)$ of a function $P : \Omega \to \mathbb{R}$ called probability, such that it holds the following postulates $P(E) \geq 0$, $P(\Omega) = 1$ and $P(E_1 \cup E_2) = P(E_1) + P(E_2)$. With the postulates it is possible to prove some probability rules such as $P(\emptyset) = 0$, $0 \leq P(E) \leq 1$ and $P(E') = 1 - P(E)$.

A *random variable* ξ is a real valued function defined over the elements of a sample space Ω, i.e., they are real numbers associated with the outcomes of an experiment. The value of random variables can be associated with an event of cardinality greater than 1. For example: If a balanced coin is tossed twice we have: $\Omega = \{ HH, HT, TH, TT \}$ with probabilities for each event of cardinality 1, $P(\{ HH \}) = 0.25$, $P(\{ HT \}) = 0.25$, $P(\{ TH \}) = 0.25$ and $P(\{ TT \}) = 0.25$. We could be interested in ξ: the number of heads obtained after the experiment, then the variable ξ can be defined as $\{ HH \} \rightarrow \xi = 2$, $\{ HT \} \rightarrow \xi = 1$, $\{ TH \} \rightarrow \xi = 1$ and $\{ TT \} \rightarrow \xi = 0$. Therefore, $\xi = 1$ can be interpreted as the set that contains elements of Ω with only one H, i.e., $\{ HT, TH \} \rightarrow \xi = 1$ and then it is clear that we can say that $P(\xi = 1) = 0.5$. Indeed, the function P is defined also for random variables, just as the example.

For the discrete case the function $F(\xi^i) = P(\xi \leq \xi^i) = \sum_{t \leq \xi^i} P(t)$, where ξ^i is a value of ξ, is called *distribution function* or the *cumulative distribution* of ξ. Furthermore, the *expectation* of a random variable ξ can be noted and computed as $E[\xi] = \sum_i \alpha^i P(\xi^i)$, where $\alpha^i = P(\xi = \xi^i)$.

5.2.2 Two-Stage Stochastic Program

A *stochastic linear program* is a linear program where some of its data is uncertain and then can be represented by random variables. If a problem has some decisions that have to be made after the uncertainty is disclosed (*realization*) it is called *recourse program* (Birge and Louveaux, 2014). As seen, the values of the random variables are known after an experiment is performed, therefore the decisions can be made in a *first stage* or in a *second stage*, before and after the experiment respectively. Furthermore, each stage could contain a sequence of decision and these can be made in different time periods. Then, a stochastic program is multi-stage if the decisions depend on realizations that occur over time.

Let $\Omega = \{ \{ \omega_1 \}, \ldots, \{ \omega_{|\Omega|} \} \}$ be a sample space. We can define a random variable ξ over the events $\{ \omega \}$ with values $\xi(\omega)$ and call any $\xi(\omega)$ as the *scenario* ω. In a stochastic program the *scenarios* can be seen in some cases as *states of the nature* as all the uncertain information could depend on them. We will suppose that it has a discrete probability distribution $f(\xi) = P(\xi = \xi(\omega)) = \alpha^\omega$.

The next program was originally proposed by Dantzig (1955) and Beale (1955):

LM 5.2.1 Two-Stage Stochastic Program with Fixed Recourse

$$\text{Minimize} \quad cx + E_\xi[\text{Min } q(\omega)y(\omega)] \tag{5.1}$$

$$\text{subject to} \quad Ax = b, \tag{5.2}$$

$$T(\omega)x + Wy(\omega) = h(\omega), \tag{5.3}$$

$$x \geq 0, \tag{5.4}$$

$$y \geq 0. \tag{5.5}$$

In LM 5.2.1 x and $y(\omega)$ are the first and second stage variables respectively. Therefore, x can be considered as decisions that are made under uncertainty and y as corrective decisions made when the uncertain is disclosed. c, b and A are vectors with appropriate dimensions. $q(\omega)$, $h(\omega)$ and $T(\omega)$ are vectors whose values will be known after the realization of ω. $T(\omega)$ is usually know as *technology matrix* and W as *recourse matrix* which is assumed to be fixed.

The following formulation is called the *deterministic equivalent program* of LM 5.2.1:

LM 5.2.2 Deterministic Equivalent Program

$$\text{Minimize} \qquad cx + \mathscr{Q}(x) \tag{5.6}$$

$$\text{subject to} \qquad Ax = b, \tag{5.7}$$

$$x \geq 0, \tag{5.8}$$

$$\text{where} \qquad \mathscr{Q}(x) = E_\xi[Q(x, \xi(\omega))] \quad \text{and} \tag{5.9}$$

$$Q(x, \xi(\omega)) = \text{Min}_y \{q(w)y \mid Wy = h(\omega) - T(\omega)x, y(\omega) \geq 0\}. \tag{5.10}$$

$\mathscr{Q}(x)$ in 5.9 is called *recourse function* and its value represents the average effect of making the decision x under the scenarios ω. $Q(x, \xi(\omega))$ in 5.10 is the *second stage program*. It shows clearly that decisions y are made after making the decisions x.

5.3 Simple Plant Location Problem with Partial Order

The Simple Plant Location Problem with Partial Order (SPLPPO) differs from the Simple Plant Location Problem with Order (SPLPO) in that the customers have preferences on a subset of facilities instead of all them. The notation and the linear model for SPLPPO are similar to those given for SPLPO, but some of them are repeated for clarity.

Let $I = \{1, \dots m\}$ be a set of customers and $J = \{1, \dots n\}$ a set of possible sites for opening facilities. Let J^i be a subset of J defined by each customer i. $|J^i| = n_i$ and $\overline{J^i} = J \setminus J^i$. Unit costs $c_{ij} \geq 0$ for supplying the demand of customer i from facility j and costs $f_j \geq 0$ for opening a facility at j are also considered. It is said that k is *i-worse* than j if customer i prefers facility j to k and it is written as $k <_i j$. We define $W_{ij} = \{k \in J \mid k <_i j\}$ as the set of facilities k strictly *i-worse* than j, its complement as $\overline{W_{ij}}$ and $W_{ij} \cup \{j\}$ as W'_{ij}. Let x_{ij} be a decision variable that represents the fraction of the demand required by customer i and supplied by facility j. Since no capacities are considered this demand will be always covered completely for one single facility, therefore optimal values x_{ij} will be in the set $\{0, 1\}$. Let y_j be a binary variable such that $y_j = 1$ if a facility is open at j and $y_j = 0$ otherwise.

Suppose that each customer i ranks its different preferred facilities $j \in J^i$ with a number $p_{ij} \in \{1, \dots, n_i\}$, where 1 and n_i are the most and the least preferred, respectively. Furthermore, for all $j \in \overline{J^i}$, $p_{ij} = n_i + 1$. Under these conditions the linear program for SPLPPO is given by LM 5.3.1. The inequalities (5.14) model the customers' preference orderings.

Note that in the case that it is more favourable to open an installation that does not belong to a particular subset J^i, the customer i can be served by any other open facility, since $n_i + 1$ is the worst rank given for any j. On the other hand, if each customer decides to be served by only one of their preferred facilities, either of the following families of constraints have to be added:

$$\sum_{k \in J^i} x_{ik} \geq y_j, \qquad \forall i \in I, \forall j \in \overline{J^i}, \tag{5.17}$$

LM 5.3.1 Simple Plant Location Problem with Partial Order

$$\text{Minimize} \quad \sum_{i\in I}\sum_{j\in J} c_{ij}x_{ij} + \sum_{j\in J} f_j y_j \tag{5.11}$$

$$\begin{aligned}
\text{subject to} \quad & \sum_{j\in J} x_{ij} = 1, && \forall i \in I, && (5.12)\\
& x_{ij} \le y_j, && \forall i \in I, \forall j \in J, && (5.13)\\
& \sum_{k\in\overline{W_{ij}}} x_{ik} \ge y_j, && \forall i \in I, \forall j \in J^i, && (5.14)\\
& x_{ij} \ge 0, && \forall i \in I, \forall j \in J, && (5.15)\\
& y_j \in \{0,1\}, && \forall j \in J. && (5.16)
\end{aligned}$$

or constraints (5.12) have to be replaced by:

$$\begin{cases} \sum_{k\in J^i} x_{ik} = 1; & \text{if } J^i \neq \emptyset, \\ \sum_{k\in J} x_{ik} = 1; & \text{if } J^i = \emptyset. \end{cases} \tag{5.18}$$

5.4 A Stochastic Formulation for SPLPPO

Consider the experiment of asking all customers i to give a rank of its preferred facilities j as mentioned before. We call each outcome of this experiment as the scenario ω and the set of all of them as the sample space Ω. We define a random matrix variable ξ with values $\xi(\omega)$ over the events $\{\omega\}$'s in Ω. We suppose that it has a discrete probability distribution $f(\xi) = P(\xi = \xi(\omega)) = \alpha^\omega$.

Under the above considerations a two-stage stochastic program for SPLPPO can be:

LM 5.4.1 Stochastic Simple Plant Location Problem with Partial Order (2S-SPLPPO)

$$\text{Minimize} \quad \sum_{j\in J} f_j y_j + \sum_{\omega\in\Omega} \alpha^\omega \sum_{i\in I}\sum_{j\in J} c_{ij}x^\omega_{ij} \tag{5.19}$$

$$\begin{aligned}
\text{subject to} \quad & \sum_{j\in J} x^\omega_{ij} = 1, && \forall i \in I, \forall \omega \in \Omega, && (5.20)\\
& x^\omega_{ij} \le y_j, && \forall i \in I, \forall j \in J, \forall \omega \in \Omega, && (5.21)\\
& \sum_{k\in\overline{W^\omega_{ij}}} x^\omega_{ik} \ge y_j, && \forall i \in I, \forall j \in J^i, \forall \omega \in \Omega, && (5.22)\\
& x^\omega_{ij} \ge 0, && \forall i \in I, \forall j \in J, \forall \omega \in \Omega, && (5.23)\\
& y_j \in \{0,1\}, && \forall j \in J. && (5.24)
\end{aligned}$$

We consider what facility to open as a first stage decision given by binary variables y's and the distribution given by x's as a second stage one. Regarding the family of constraints (5.22), each scenario ω is represented by a $\{0,1\}$-matrix $W(\omega)$ ($\overline{W(\omega)}$) of dimension $(mn \times mn)$, where each row ij has 1 for each element of W_{ij}^{ω} ($\overline{W_{ij}^{\omega}}$) and 0 for elements of its complement. See Figure 5.1. Furthermore, Ω is a finite set but can have a very big cardinality. For instance if we consider no partial preferences and since $n < +\infty$, each row of $W(\omega)$ can be given in 2^n different ways, therefore there are $(2^n)^{mn} = 2^{mn^2}$ possible different matrices $W(\omega)$.

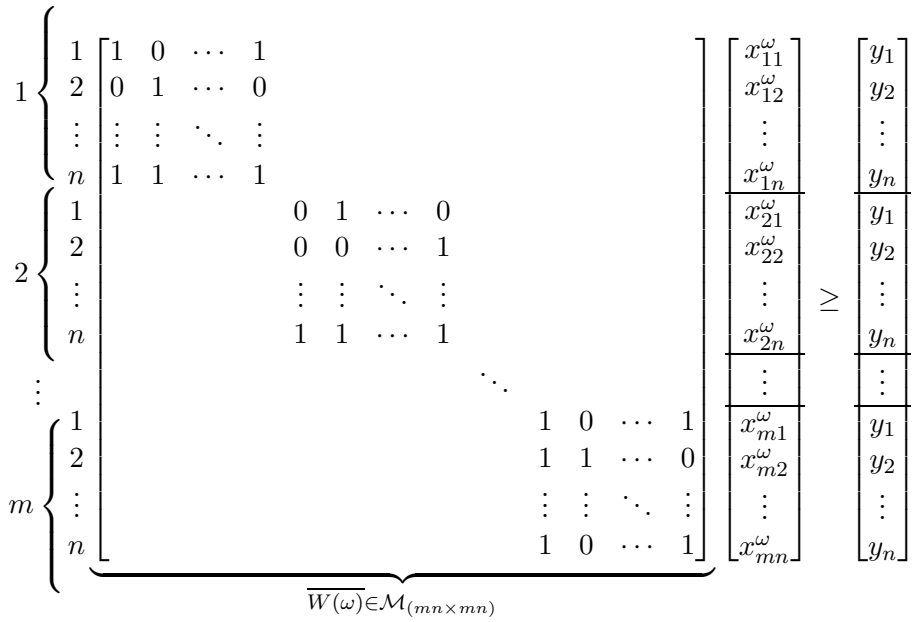


Figure 5.1: Example $\overline{W(\omega)}$.

The stochastic program has no fixed resources since the coefficient matrix $\overline{W(\omega)}$ given by the sets $\overline{W_{ij}^{\omega}}$ with the second-stage variables x_{ij}^{ω} is random, i.e., it depends of ω.

The second-stage program is given by:

$$Q(y, \xi(\omega)) = \text{Min}_x \left\{ \sum_{i \in I} \sum_{j \in J} c_{ij} x_{ij}^{\omega} \mid (5.20), (5.21), (5.22), (5.23) \right\},$$

and if $\mathscr{Q}(y) = E_{\xi}[Q(y, \xi(\omega))]$ we obtain the deterministic equivalent program:

$$\text{Min} \left\{ \sum_{j \in J} f_j y_j + \mathscr{Q}(y) \mid (5.24) \right\},$$

where E_{ξ} is the mathematical expectation on the random vector values ξ.

5.5 Computational Results for 2S-SPLPPO

We carried out experiments to determine how strong is the influence of uncertainty in 2S-SPLPPO when the rank of preferences of each customer is modified in a given percentage from its original values. We considered problems with $m = 75$ customers and $n = 50$ facilities with complete preferences and two scenarios (ω_1 and ω_2) for each. Then, we compare the objective function OF_S and time t_S when running a whole program with the results of running a problem for every scenario and we check how different are the open facilities on each case.

If we supposed that the preferences of customers are known and repeated periodically, then the average in the long run is given by $(\text{OF}_{\omega_1} + \text{OF}_{\omega_2})/2$ and the loss of benefit on OF due the presence of uncertainty, i.e., the expected value of perfect information EVPI, is given by the difference between OF of the two-stage stochastic program and this average with perfect forecast:

$$\text{EVPI} = \text{OF}_S - \left(\frac{\text{OF}_{\omega_1} + \text{OF}_{\omega_2}}{2}\right)$$

.

Similarly, the percentage of lost time over the total used time due to stochasticity, i.e., expected lost time of perfect information ELTPI. It can be computed by:

$$\text{ELTPI} = \left(t_S - \left(\frac{t_{\omega 1} + t_{\omega 2}}{2}\right)\right) \Big/ t_S$$

.

The headers of the tables have the following meanings:

- OF: Optimal objective function value.
- y: Number of opened facilities.
- t: CPU time in seconds.
- avg: Average.
- EVPI: Expected value of perfect information.
- ELTPI(%): Expected lost time of perfect information.
- y-dif: Manhattan distance between y1 and y2.

In Tables 5.1, 5.2, 5.3 and 5.4 the rank of preferences in ω_2 were modified by replacing a $p\%$ of the most preferred facilities for each customer i in ω_1 with other facilities among the remaining non-preferred ones.

Table 5.1: Computational results (10% of ω_2 of most preferred from ω_1 has been changed).

				Scenarios											
	Stochastic			ω_1			ω_2			avg		ELTPI	y-dif		
Problem	OF	y	t	OF	y	t	OF	y	t	OF	EVPI	%	S-ω1	S-ω2	ω1-ω2
10w2a7550_1	1648853	7	67	1661269	7	16	1636437	7	24	1648853	0	70%	0	0	0
10w2a7550_2	1602647	4	48	1632213	7	18	1571827	4	23	1602020	627	57%	5	0	5
10w2b7550_1	1226979	8	48	1252804	8	11	1195753	9	16	1224279	2701	72%	0	1	1
10w2b7550_2	1263465	11	53	1249750	9	17	1197943	11	15	1223847	39619	69%	10	8	16
10w2c7550_1	1290291	14	46	1310193	11	17	1257497	14	19	1283845	6446	62%	9	2	11
10w2c7550_2	1248312	11	47	1201706	12	8	1175249	14	13	1188478	59835	77%	5	15	12

Table 5.2: Computational results (25% of ω_2 of most preferred from ω_1 has been changed).

	Stochastic			Scenarios ω_1			ω_2			avg		ELTPI	y-dif		
Problem	OF	y	t	OF	y	t	OF	y	t	OF	EVPI	%	S-ω1	S-ω2	ω1-ω2
25w2a7550_1	1670734	5	119	1661269	7	16	1614826	7	26	1638048	32687	82%	6	8	12
25w2a7550_2	1606360	4	67	1632213	7	18	1548535	8	18	1590374	15986	73%	9	10	5
25w2b7550_1	1256755	6	58	1252804	8	11	1164725	11	17	1208765	47991	76%	2	13	13
25w2b7550_2	1324578	7	153	1249750	9	17	1217989	12	18	1233870	90709	88%	10	13	13
25w2c7550_1	1364933	11	166	1310193	11	17	1226890	11	15	1268542	96392	90%	16	8	14
25w2c7550_2	1271029	15	84	1201706	12	8	1192867	15	16	1197287	73743	85%	7	10	13

Table 5.3: Computational results (50% of ω_2 of most preferred from ω_1 has been changed).

	Stochastic			Scenarios ω_1			ω_2			avg		ELTPI	y-dif		
Problem	OF	y	t	OF	y	t	OF	y	t	OF	EVPI	%	S-ω1	S-ω2	ω1-ω2
50w2a7550_1	1689428	4	113	1661269	7	16	1599711	6	23	1630490	58938	82%	7	4	9
50w2a7550_2	1637084	4	86	1632213	7	18	1570431	6	18	1601322	35762	79%	5	6	9
50w2b7550_1	1331207	7	165	1252804	8	11	1166528	10	14	1209666	121541	92%	9	9	12
50w2b7550_2	1307706	9	182	1249750	9	17	1131067	11	16	1190409	117298	91%	12	12	18
50w2c7550_1	1347482	7	145	1310193	11	17	1227289	11	15	1268741	78741	89%	14	10	16
50w2c7550_2	1287982	12	112	1201706	12	8	1118653	10	11	1160180	127803	92%	6	12	16

Table 5.4: Computational results (100% of ω_2 of most preferred from ω_1 has been changed).

	Stochastic			Scenarios ω_1			ω_2			avg		ELTPI	y-dif		
Problem	OF	y	t	OF	y	t	OF	y	t	OF	EVPI	%	S-ω1	S-ω2	ω1-ω2
100w2a7550_1	1787955	1	267	1661269	7	16	1632907	6	21	1647088	140867	93%	8	7	13
100w2a7550_2	1683058	4	194	1632213	7	18	1585028	5	18	1608621	74438	91%	5	5	8
100w2b7550_1	1451139	9	363	1252804	8	11	1337446	9	20	1295125	156014	96%	7	16	13
100w2b7550_2	1400184	8	271	1249750	9	17	1217508	9	12	1233629	166555	95%	13	9	14
100w2c7550_1	1360674	10	169	1310193	11	17	1244255	10	11	1277224	83450	92%	9	6	13
100w2c7550_2	1402514	11	196	1201706	12	8	1334782	11	21	1268244	134270	92%	7	10	13

As can be noticed, the value of EVPI seems to increase as the percentage of changed preferences increases. ELTPI is high in all cases, but a slight increase can be seen in the last tables. The results suggest that there is no dependence between the percentage of changed preferred facilities and the open facilities in the solutions on each scenario and the stochastic problem. Additional examples can be seen in Appendix A.

Since the 2S-SPLPPO formulation (see LM 5.4.1) has the same characteristics of the SPLPO with evident differences due to the presence of multiple scenarios, we suggest to use the ADA algorithm proposed in Section 4 to solve it.

Indeed, by following the same steps given in Sections 4.3 and 4.4 we obtain a lagrangean and semi-lagrangean relaxation version to SPLPPO:

A Lagrangean Relaxation for 2S-SPLPPO:

$$LR(\mu^\omega, \lambda^\omega) = \min_{(x^\omega, y)} \sum_j \left[\sum_\omega \sum_i \left(\alpha^\omega c_{ij} - \mu_i^\omega - \Lambda_{ij}^\omega \right) x_{ij}^\omega + \left(f_j + \sum_\omega \sum_i \lambda_{ij}^\omega \right) y_j \right] + \sum_\omega \sum_i \mu_i^\omega,$$

subject to: (5.21), (5.23) and (5.24).

where $\Lambda_{ij}^\omega = \sum_{\substack{k \in W_{ij}'^\omega \\ |W_{ij}'^\omega| = n - p_{ij}^\omega + 1}} \lambda_{ik}^\omega$.

A Semi-Lagrangean Relaxation for 2S-SPLPPO:

$$SLR(\gamma^\omega) = \min_{(x^\omega, y)} \sum_j \left(\sum_\omega \sum_i \left(\alpha^\omega c_{ij} - \gamma_i^\omega \right) x_{ij}^\omega + f_j y_j \right) + \sum_\omega \sum_i \gamma_i^\omega,$$

subject to: (5.21), (5.22), (5.23), (5.24), and $\sum_{j \in J} x_{ij}^\omega \leq 1$.

Moreover, it is not difficult to extend Theorems 14, 15 and 16 to the 2S-SPLPPO case.

In Table 5.5 we present the results after applying ADA to the same problems given in Tables 5.1, 5.2, 5.3 and 5.4. The parameters for these experiments are the same as those used for instances ($m = 75$, $n = 50$): $sg\_iter = 50$, $da\_iter = 3$ and $fhv\_iter = 2$. Also, the header of the tables remains the same:

- Prob: Name of the problem.
- Optimal: Optimal objective function value of the problem P.
- LP(P): Linear relaxation value for problem P.
- y_j: Number of opened facilities.
- t (Tt): CPU time in seconds (Total time).
- bestUB: best upper bound.
- SG: Best value using the subgradient method for $LD_{\mu\lambda}$.
- UB (LB): Upper bound (Lower bound).
- GAP_o (%): bestUB-Opt ($\frac{\text{bestUB-Opt}}{\text{Opt}} \times 100\%$). The absolute and relative gap between the optimal and the value of an algorithm.

The results show that ADA performs very well in all cases. Indeed, the optimum was founded in most of them. It can be observed that when the percentage of changed preferences was 10% (first six instances) ADA was not able to improve the Xpress times. However, in the remaining cases the times were improved considerably with a couple of exceptions.

Table 5.5: ADA applied to 2S-SPLPPO with two scenarios. ($m = 75$, $n = 50$).

	Xpress				Hc			SG		DA with VFH				ADA		
Prob	**Optimal**	**LP**	**y_j**	**t**	**bestUB**	**y_j**	**t**	**LB**	**t**	**bestUB**	**LB**	**y_j**	**t**	**Tt**	**imp t**	**GAP**
10w2a7550_1	1648853	1200745	7	67	1787955	1	0	840831	7	1698650	1279041	4	60	67	0%	3.02%
10w2a7550_2	1602647	1193378	4	48	1738071	3	0	763664	6	1602647	1286736	4	50	56	-17%	0.00%
10w2b7550_1	1226979	901051	8	48	1335244	10	0	334589	5	1226979	953159	8	57	62	-28%	0.00%
10w2b7550_2	1263465	906280	11	53	1299259	10	0	372981	7	1271641	970770	10	70	77	-46%	0.65%
10w2c7550_1	1290291	900033	14	46	1388280	9	0	591938	7	1290291	971785	14	62	69	-48%	0.00%
10w2c7550_2	1248312	876053	11	47	1365717	13	0	557308	6	1248312	936483	11	65	71	-51%	0.00%
25w2a7550_1	1670734	1199273	5	119	1761050	5	0	830624	6	1670734	1284052	5	53	59	50%	0.00%
25w2a7550_2	1606360	1195482	4	67	1730389	3	0	770870	7	1606360	1269850	4	53	60	11%	0.00%
25w2b7550_1	1256755	892106	6	58	1329801	11	0	360026	6	1259217	947822	8	61	67	-15%	0.20%
25w2b7550_2	1324578	907308	7	153	1348842	7	0	380921	6	1324578	988646	7	98	104	32%	0.00%
25w2c7550_1	1364933	907957	11	166	1413489	9	0	579362	8	1373542	978857	14	96	104	37%	0.63%
25w2c7550_2	1271029	873435	15	84	1291327	13	0	578313	7	1271029	958058	15	77	84	0%	0.00%
50w2a7550_1	1689428	1212042	4	113	1778238	6	0	804221	7	1689428	1302236	4	60	67	41%	0.00%
50w2a7550_2	1637084	1196440	4	86	1744372	5	0	760256	6	1637084	1273267	4	51	58	33%	0.00%
50w2b7550_1	1331207	900654	7	165	1388062	12	0	294443	7	1331207	1247451	7	381	388	-136%	0.00%
50w2b7550_2	1307706	896181	9	182	1364303	11	0	305591	7	1307706	990369	9	94	101	45%	0.00%
50w2c7550_1	1347482	911370	7	145	1446345	12	0	565472	6	1347482	976268	7	77	84	42%	0.00%
50w2c7550_2	1287982	878852	12	112	1439845	17	0	546735	7	1299952	952655	12	71	78	30%	0.93%
100w2a7550_1	1787955	1208842	1	267	1787955	1	0	824731	8	1787955	1293098	1	89	97	64%	0.00%
100w2a7550_2	1683058	1204739	4	194	1737924	5	0	827668	7	1720482	1268026	4	66	72	63%	2.22%
100w2b7550_1	1451139	935113	9	363	1496648	6	6	311032	6	1453678	989340	6	86	92	75%	0.17%
100w2b7550_2	1400184	916167	8	271	1449128	9	0	358259	7	1400205	1013362	10	106	113	58%	0.00%
100w2c7550_1	1360674	920342	10	169	1430298	9	0	562495	8	1360674	998681	10	106	114	32%	0.00%
100w2c7550_2	1402514	918696	11	196	1462990	16	0	570058	8	1404147	989592	11	102	110	44%	0.12%

Table 5.6: ADA applied to 2S-SPLPPO with two scenarios.

	Xpress				Hc			SG		DA with VFH				ADA		
Prob	**Optimal**	**LP**	**y_j**	**t**	**bestUB**	**y_j**	**t**	**LB**	**t**	**bestUB**	**LB**	**y_j**	**t**	**Tt**	**imp t**	**GAP**
100w2a10075_1	2469439	1811464	2	561	2476632	1	1	1644492	31	2476632	1978083	3	265	296	47%	0.29%
100w2a10075_2	2458870	1805025	3	685	2476632	1	1	1627278	32	2458870	1971011	3	237	270	61%	0.00%
100w2b10075_1	2132719	1364985	5	17935	2270467	6	1	1112051	37	2132719	1558555	5	3531	3568	80%	0.00%
100w2b10075_2	2163818	1367450	7	27679	2218215	7	1	1160875	43	2163818	1564516	7	4380	4422	84%	0.00%
100w2c10075_1	1978807	1271848	7	14835	2072702	6	1	1052860	49	1988903	1496210	7	1027	1076	93%	0.51%
100w2c10075_2	1987757	1261290	6	11567	2118928	9	1	1066388	51	1987757	1452609	6	3152	3202	72%	0.00%
100w2a125100_1	3070535	2416518	1	918	3070535	1	2	2237118	93	3070535	2619571	1	573	666	27%	0.00%
100w2a125100_2	3070535	2388054	1	1088	3070535	1	1	2239726	106	3070535	2587669	1	702	808	26%	0.00%
100w2b125100_1	2800573	1815018	8	53666	2850413	5	2	1601481	118	2850413	2078979	7	20837	20955	61%	1.78%
100w2b125100_2	2820883	1820001	5	78669	3019740	4	1	1592305	143	2820883	2016632	5	8230	8373	89%	0.00%
100w2c125100_1	2702169	1698737	9	239967	2866218	10	2	1488148	157	2702169	1990068	9	23717	23874	90%	0.00%
100w2c125100_2	2716252	1705149	6	204007	2829945	5	1	1477796	168	2717597	2007831	7	27442	27610	86%	0.05%
100w2a150100_1	3768087	2924250	1	1735	3768087	1	1	2699949	109	3768087	3239975	1	905	1014	42%	0.00%
100w2a150100_2	3768087	2918397	1	1819	3768087	1	1	2702231	111	3768087	3251719	1	117	228	87%	0.00%
100w2b150100_1	3412417	2179897	6	169739	3637438	1	1	1923456	141	3412417	2599980	6	21111	21252	87%	0.00%
100w2b150100_2	3388309	2196284	3	69508	3637438	1	2	1924300	169	3388309	2679093	3	14792	14962	78%	0.00%
100w2c150100_1	3287595	2010587	5	502354	3413288	4	2	1768341	185	3413288	-	-	-	-	-	3.82%
100w2c150100_2	3229424	2012045	6	494721	3307459	5	3	1776475	132	3300341	2474515	7	119700	119832	76%	2.20%
100w2a150100_1	3768087	2924250	1	1735	3768087	1	1	2635877	94	3768087	3229888	1	1159	1253	28%	0.00%
100w2a150100_2	3768087	2918397	1	1819	3768087	1	1	2637900	109	3768087	3242679	1	1255	1364	25%	0.00%
100w2b150100_1	3412417	2179897	6	169739	3637438	1	2	1876775	147	3445585	2607746	4	21590	21738	87%	0.97%
100w2b150100_2	3388309	2196284	3	69508	3637438	4	4	1863975	155	3388309	2599780	3	8984	9139	87%	0.00%
100w2c150100_1	3287595	2010587	5	502354	3413288	4	2	1750243	179	3288348	2457997	6	78577	78756	84%	0.02%
100w2c150100_2	3229424	2012045	6	494721	3307459	5	3	1718731	185	3230261	2470331	7	144729	144729	71%	0.03%

We have applied ADA to bigger instances ($m = 100$, $n = 75$), ($m = 125$, $n = 100$) and ($m = 150$, $n = 100$). For the first three groups, the parameters used were:

- Group 1 ($m = 100$,$n = 75$): $sg\_iter = 100$, $da\_iter = 7$ and $fhv\_iter = 2$
- Group 2 ($m = 125$,$n = 100$): $sg\_iter = 170$, $da\_iter = 10$ and $fhv\_iter = 2$
- Group 3 ($m = 150$,$n = 100$): $sg\_iter = 170$, $da\_iter = 12$ and $fhv\_iter = 2$

The result can be seen in Table 5.6. ADA met the optimal solution in most of the cases in much less time, with the exception of 100w2c150100_1 where the time was bigger than this obtained by Xpress. The last of the four groups corresponds to the same problems in Group 3 ($m = 150$, $n = 100$), but they were tested with a different parameter $sg\_iter = 140$. This can be seen for all these six cases that ADA improves the running times and bounds.

5.6 Conclusions

We have defined a more general model for SPLPO where the order can be partial. Also, a stochastic formulation is presented with the order being considered a random variable. Regarding the solution of this model, we have found as the scenarios in the stochastic model differ with each other, the solving time increases. Furthermore, it has been possible to determine that the stochastic version 2S-SPLPPO shares the same properties as those proved for SPLPO, and therefore the ADA algorithm 4.5.2 can be applied for this case. The computational results show that ADA performs satisfactorily on large instances, both in the search of the optimum and in the execution time. A parameter setting studies and experiments with more than two scenarios must be carried out.

Chapter 6

Final Conclusions, Remarks and Future Work

Two discrete optimization problems arising from real application have been studied in this thesis: Traffic Light Synchronization (TLSP) and The Simple Plant Location with Order (SPLPO). We have also studied other problems related to them to extend their applicability.

Even though the bandwidth maximization formulation for TLSP has been widely studied, this is not the most used approach in real-world applications. Most of the commercial software base their solution methods on formulations that minimize other measures of interest as the total delay time of vehicles, with very good results. In spite of this, the interest in the bandwidth approach has remained. Some recent publications generalize the MAXBAND model thus improving previous results for the arterial case, but studies of their applications to large instances in networks are not so popular. In Chapter 2 we address this case. The systematic classic algorithm in Section 2.5 solves efficiently a particular case of bandwidth maximization problem, the arterial case with common period. It is clear that it can be solved via linear programming by removing network constraints in MAXBAND formulation. However, in addition to its historical importance, the study of this algorithm has led us to a better understanding of the geometry which is useful to build arterial and network constraints. On the other hand, in Section 2.6 we delved into the whole MAXBAND model and extended the bounds for integer variables given by the single arterial case to networks. We were able to produce a heuristic algorithm based on tabu search that takes advantage of the mixed integer linear MAXBAND model to obtain feasible solutions. We obtained good results in larger instances than those known in the literature.

A future work must consider other aspects that our algorithm does not take into account, such as prioritizing arteries with a high vehicular flow. Additionally, finding an initial solution by a heuristic method still remains pending. Waiting for a feasible solution from branch and bound is, in some cases, very expensive in terms of running times. The traffic light synchronization is a kind of problem where a single feasible solution could be enough to be accepted as a "good" solution, as the perfect synchronization could not be possible. Therefore, an initial systematic algorithm can be in fact the procedure that we have been trying to find. However, our results showed that when the branch and bound method is able to find the initial starting solution quickly, its bandwidth can be overcome in a reasonable time. We thought that the results could be also validated if performance indices are obtained on real-world data comparable to those obtained by commercial software such as TRANSYT. Unfortunately, we could not access this software or instances already tested on it. In addition, we mention that to our knowledge there is no polyhedral study on the MAXBAND constraints. This is important since could shed light on the development of possible exact methods of solution.

A related problem to TLSP was studied in Chapter 3, The Shortest Path Problem with Traffic Light Constraints (SPPTL). We proposed a linear model which is a generalization of the flow-based formulation for The Shortest Path Problem (SPP). The linear program models the behaviour of traffic lights through time with periodic time windows. As far as we know, there is not a previous linear model to this problem in literature.

Is this the best way to formulate the model? It is a question that remains to be solved. As mentioned in Section 3.4, our future work aims to improve the proposed formulation by studying different ways of representing the problem and one of them is by using a time-space network scheme. Although this approach has been successfully applied to scheduling problems, some vehicle routing applications can be found in the literature. In our opinion, this scheme adjusts to the SPPTL problem since it requires the use of periodic time windows.

As was mentioned in previous chapters, there are very few studies available for The Simple Plant Location Problem with Order (SPLPO). The main results available face the problem by studying its linear model to add new valid constraints to tighten the formulation. This led to the development of exact methods. However, the particular case without order, the Simple Plant Location Problem (SPLP), has been studied much more. Particularly, in Chapter 4 we were interested in the theoretical results of Lagrangean and semi-Lagrangean relaxation for this case. We proved that those results can be extended to SPLPO. We proposed an algorithm (ADA) that exploit these ideas and that performed very well on large instances, finding the optimal in most cases tested.

Regarding this case, we want to point out that we still have the task of carrying out a parameter calibration. In this work an empirical setting was made by repeated experiments. In addition, we are interested in verifying if a similar approach can be applied to other location problems, e.g., p-median problems with customer preferences. Problems of this type without preferences have already been studied successfully using semi-Lagrangean relaxation.

In Chapter 5 we provided a more general formulation where the order can be partial (SPLPPO) and then introduced a stochastic version (2S-SPLPPO). We were able to apply the same procedure to 2S-SPLPPO. Again, the optimal was found in most of the instances tested.

The last results are promising, but we are aware that deeper studies on parameter settings must be made, as well as an analysis of stopping criterion for Lagrangean and semi-Lagrangean procedures. Our experiments confirmed that a large number of iterations in both procedures lead to an increase in running times even worse than those obtained by Xpress. We are currently working on evaluating our procedure using problems with a larger number of scenarios.

Bibliography

Balakrishnan, N. (1993). Simple heuristics for the vehicle routing problem with soft time windows. *Journal of Operational Research Society*, 44:279–287.

Beale, E. (1955). On minimizing a convex function subject to linear inequalities. *Journal of the Royal Statistical Society, Series B*, 17:173–184.

Beasley, E. (1990). OR-library: Distributing test problems by electronic mail. *Journal of the Operational Research Society*, 41(11):1069–1072.

Beltrán, C., Tandoki, C., and Vial, J. (2006). Solving the p-median problem with a semi-Lagrangian relaxation. *Computational Optimization and Applications*, 35:239–260.

Beltrán, C., Vial, J., and Alonso, A. (2012). Semi-Lagrangian relaxation applied to the uncapacited facility location problem. *Computational Optimization and Applications*, 51:387–409.

Benders, J. (1962). Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, 4:238–252.

Bilde, O. and Krarup, J. (1977). Sharp lower bounds and efficient algorithms for the simple plant location problem. *Annals of Discrete Mathematics*, 1:79–97.

Birge, J. and Louveaux, F. (2014). *Introduction to stochastic programming*. Series in Operations Research and Finantial Engineering, Springer, 2nd edition.

Braun, R. and Weichenmeier, F. (2005). Automatic offline-optimization of coordinated traffic signal control in urban networks using genetic algorithms. *Proceedings of the 12th World Congress on Intelligent Transport Systems*.

Cánovas, L., García, S., Labbé, M., and Marín, A. (2006). A strengthened formulation for the simple plant location problem with order. *Operations Research Letters*, 35:141–150.

Chaudhary, N. (1987). *A mixed integer linear programming approach for obtaining and optimal signal timing plan in general traffic networks*. PhD thesis, Texas A&M University.

Chen, Y. and Yang, H. (2000). Shortest paths in traffic-light networks. *Transportation Research Part B*, 34(4):241–253.

Cohen, S. (1983). Concurrent use of MAXBAND and TRANSYT signal timing programs for arterial signal optimization. *Transportation Research Record*, 906:81–84.

Conforti, M., Cornuéjols, G., and Zambelli, G. (2014). *Integer programming*. Springer.

Cornuéjols, G., Fisher, M., and Nemhauser, G. (1977). Location of bank accounts to optimize float: an analytic study of axact and approximated algorithms. *Management Science*, 23(8):789–810.

Dantzig, G. (1955). Linear programming under uncertainty. *Management Science*, 1:197–206.

Diestel, R. (2000). *Graph theory*. Graduate Texts in Mathematics, Springer, 3rd edition.

Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271.

Erlenkotter, D. (1978). A dual-based procedure for uncapacitated facility location. *Operations Research*, 26(6):992–1009.

Fisher, M. L. (2004). The Lagrangian relaxation method for solving integer programming problems algorithms. *Management Science*, 50(12):1861–1871.

Fredman, M. L. and Tarjan, R. E. (1987). Fibonacci heaps and their uses in improved network optimization algorithms. *Journal of the ACM (JACM)*, 34(3):596–615.

Freund, J., Miller, I., and Miller, M. (2014). *Mathematical Statistics with Applications*. Pearson, 8th edition.

Gartner, N., Assmann, S., Lasaga, F., and Hou, D. (1991). A multi-band approach to arterial traffic signal. *Transportation Research*, 25B(1):55–74.

Gartner, N., Little, D., and Gabbay, H. (1975). Optimization of traffic signal settings by mixed-integer linear programming; part I: The network coordination problem; part II: The network synchronization problem. *Transportation Science*, 9:321–363.

Gartner, N. and Stamatiadis, C. (2002). Arterial-based control of traffic flow in urban grid networks. *Mathematical and Computer Modelling*, 35:657–671.

Geoffrion, A. (1974). Lagrangian relaxation for integer programming. *Mathematical Programming Study*, 2:82–114.

Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computers and Operations Research*, 13:533–549.

Guignard, M. (2003). Lagrangean relaxation. A tutorial. *TOP*, 11(2):151–228.

Guignard, M. and Opaswongkarn, K. (1990). Lagrangean dual ascent algorithms for computing bounds in capacitated plant location problems. *European Journal of Operational Research*, 46(1):73–83.

Hane, C. A., Barnhart, C., Johnson, E., Marsten, R., Nemhauser, G., and Sigismondi, G. (1995). The fleet assignment problem: Solving a large-scale integer program. *Mathematical Programming*, 70(1):211–232.

Hanjoul, P. and Peeters, D. (1987). A facility location problem with clients preference orderings. *Regional Science and Urban Economics*, 17:451–473.

Held, M. and Karp, R. (1971). The traveling salesman problem and minimum spanning trees: part II. *Mathematical Programming*, 1:6–25.

Held, M., Wolfe, P., and Crowder, H. (1974). Validation of subgradient optimization. *Mathematical Programming*, 6:62–88.

Hotovy, R., Larson, D., and Scholze, S. (2015). Binary frames. *Houston Journal of Mathematics*, 41(3).

Improta, G. and Sforza, A. (1982). Optimal offsets for traffic signal systems in urban networks. *Transportation Research Part B: Methodological*, 16(2):143–161.

Jörnsten, K. (2016). An improved Lagrangian relaxation and dual ascent approach to facility location problems. *Computational Management Science*, 13:317–348.

Kall, P. and Mayer, J. (2005). *Stochastic linear programming: Models, Theory, and Computation*. Springer's international series.

Kavitha, T., Liebchen, C., Mehlhorn, K., Michail, D., Rizzi, R., Ueckerdt, T., and Zweig, K. (2009). Cycle bases in graphs: Characterization, algorithms, complexity and applications. *Computer Science*, 3(4):199–243.

Kirkpatrick, L., Gelatt Jr., C., and Vecchi, M. (1983). Optimization by simulated annealing. *Procedia - Social and Behavioral Sciences*, 220(4598):671–680.

Kliewer, N., Mellouli, T., and Suhl, L. (2006). A timespace network based exact optimization model for multi-depot bus scheduling. *European Journal of Operational Research*, 175(3):1616–1627.

Köhler, E. and Strehler, M. (2015). Traffic signal optimization using cyclically expanded networks. *Wiley Periodicals, Networks*, 65:244–261.

Laporte, G. and Louveaux, F. (1993). The integer l-shaped method for stochastic integer programs with complete recourse. *Operations Research Letters*, 13:133–142.

Liebchen, C. and Rizzi, R. (2007). Classes of cycle bases. *Discrete Applied Mathematics*, 155(3):337–355.

Little, D. (1966). The synchronization of traffic signals by mixed-integer linear programming. *Operations Research*, 14(4):568–594.

Little, D., Kelson, M., and Gartner, N. (1981). MAXBAND: A versatile program for setting signal on arteries and triangular networks. *Transportation Research Record*, 795:40–46.

Lu, T., Sohr, A., and Bei, X. (2014). Comparison of the effectiveness of common cycle computing models. *Procedia - Social and Behavioral Sciences*, 138:358–367.

MacLane, S. (1937). A combinatorial condition for planar graphs. *Fundamental Mathematicae*, 28:22–32.

Mahmoudi, M. and Zhou, X. (2016). Finding optimal solutions for vehicle routing problem with pickup and delivery services with time windows: A dynamic programming approach based on statespacetime network representations. *Transportation Research Part B: Methodological*, 89:19–42.

Mesquita, M., Murta, A., Paias, A., and Wise, L. (2013). TSP with multiple time-windows and selective cities. *Computational Logistics: 4th International Conference, ICCL 2013, Copenhagen, Denmark, September 25-27, 2013. Proceedings*, pages 158–172.

Mladenović, N. and Hansen, P. (1997). Variable neighbourhood search. *Computers and Operations Research*, 24:1097–1100.

Monabbati, E. (2014). An application of a Lagrangian-type relaxation for the uncapacitated facility location problem. *Japan Journal of Industrial and Applied Mathematics*, 31:483–499.

Morgan, J. and Little, D. (1964). Synchronizing traffic signals for maximal bandwidth. *Operations Research*, 12(6):896–912.

Poljak, B. T. (1967). A general method for solving extremum problems. *Soviet Mathematics Doklady*, 8:593–597.

Ratrout, N. and Reza, I. (2014). Comparison of optimal signal plans by synchro and transyt-7f using paramics-a case study. *Procedia Computer Science*, 32:372–379.

Robertson, D. (1969). *TRANSYT, a traffic network study tool. Technical Report.* Crowthorne, Berkshire.

Russell, R. (1995). Hybrid heuristics for the vehicle-routing problem with time windows. *Transportation Sciences*, 29:156–166.

Schrijver, A. (1986). *Theory of linear and integer programming.* John Wiley & Sons, 1st edition.

Singh, L., Tripathi, S., and Arora, H. (2009). Time optimization for traffic signal control using genetic algorithm. *Expert Systems with Applications*, 2(2).

Vasilyev, I. and Klimentova, X. (2010). The branch and cut method for the facility location problem with client's preferences. *Journal of Applied and Industrial Mathematics*, 4(3):441–454.

Vasilyev, I. L., Klimentova, X., and Boccia, M. (2013). Polyhedral study of simple plant location problem with order. *Operations Research Letters*, 41:153–158.

Wollenweber, J. (2008). A multi-stage facility location problem with staircase costs and splitting of commodities: model, heuristic approach and application. *OR Spectrum*, 30(4):655–673.

Wünsch, G. (2008). *Coordination of traffic signals in networks.* PhD thesis, Technische Universität Berlin.

Xianyu, W., Peifeng, H., and Zhenzhou, Y. (2013). Link-based signalized arterial progression optimization with practical travel speed. *Journal of Applied Mathematics.*

Xianyu, W., Zong, T., Peifeng, H., and Zhenzhou, Y. (2012). Bandwidth optimization of coordinated arterials based on group partition method. *Procedia - Social and Behavioral Sciences*, 43:232–244.

Zhang, C., Xie, Y., Gartner, N., Stamatiadis, C., and Arsava, T. (2015). AM-BAND: An asymmetrical multi-band model for arterial traffic signal coordination. *Transportation Research Part C: Emerging Technologies*, 58:515–531.

Appendix A

Aditional Computational Experiments for 2S-SPLPPO

In the next tables the rank of preferences in ω_3 are a random permutation of the rank preference in ω_1. The conclusions remains similar to those in Section 5.5.

Table A.1: Computational results (10% of ω_3 of most preferred from ω_1 has been changed).

				Scenarios											
	Stochastic			ω_1			ω_3			avg		ELTPI		y-dif	
Problem	OF	y	t	OF	y	t	OF	y	t	OF	EVPI	%	S-ω1	S-ω3	ω1-ω3
10w3a7550_1	1650081	5	72.587	1661269	7	16	1637285	5	20.92	1649277	804	74%	6	0	6
10w3a7550_2	1609486	4	55.848	1632213	7	18	1585505	4	14.43	1608859	627	71%	5	0	5
10w3b7550_1	1296049	8	63.82	1252804	8	11	1295936	7	19.859	1274370	21679	76%	4	9	7
10w3b7550_2	1257586	9	39.219	1249750	9	17	1265336	10	19.781	1257543	43	53%	0	3	3
10w3c7550_1	1353045	9	81.557	1310193	11	17	1332265	9	25.865	1321229	31816	74%	10	0	10
10w3c7550_2	1300803	11	50.186	1201706	12	8	1276016	12	15.709	1238861	61942	76%	15	5	10

Table A.2: Computational results (25% of ω_3 of most preferred from ω_1 has been changed).

				Scenarios											
	Stochastic			ω_1			ω_3			avg		ELTPI		y-dif	
Problem	OF	y	t	OF	y	t	OF	y	t	OF	EVPI	%	S-ω1	S-ω3	ω1-ω3
25w3a7550_1	1694543	4	61	1661269	7	16	1673216	6	16	1667243	27301	73%	5	2	7
25w3a7550_2	1650736	4	70	1632213	7	18	1612348	4	17	1622281	28456	76%	11	0	11
25w3b7550_1	1312507	7	75	1252804	8	11	1324685	6	17	1288745	23763	81%	5	7	4
25w3b7550_2	1374306	4	90	1249750	9	17	1380846	5	22	1315298	59008	78%	11	3	14
25w3c7550_1	1461222	9	163	1310193	11	17	1478011	9	22	1394102	67120	88%	14	8	10
25w3c7550_2	1436945	11	168	1201706	12	8	1458695	9	28	1330201	106745	89%	7	12	11

Table A.3: Computational results (50% of ω_3 of most preferred from ω_1 has been changed).

				Scenarios											
	Stochastic			ω_1			ω_3			avg		ELTPI	y-dif		
Problem	OF	y	t	OF	y	t	OF	y	t	OF	EVPI	%	S-ω1	S-ω3	ω1-ω3
50w3a7550_1	1767341	3	74	1661269	7	16	1741438	3	15	1701354	65988	79%	6	2	8
50w3a7550_2	1682656	2	43	1632213	7	18	1700286	2	23	1666250	16407	53%	9	0	9
50w3b7550_1	1460933	7	178	1252804	8	11	1463203	8	18	1358004	102930	92%	3	9	10
50w3b7550_2	1455769	5	155	1249750	9	17	1497071	2	22	1373411	82359	87%	12	5	11
50w3c7550_1	1541597	6	206	1310193	11	17	1594566	5	25	1452380	89218	90%	13	1	14
50w3c7550_2	1510410	11	212	1201706	12	8	1533338	7	23	1367522	142888	92%	13	12	17